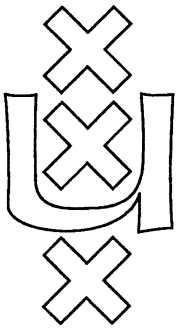
Institute for Language, Logic and Information

KOHONEN FEATURE MAPS IN NATURAL LANGUAGE PROCESSING

J.C. Scholtes

ITLI Prepublication Series
for Computational Linguistics CL-91-01



University of Amsterdam

The ITLI Prepublication Series

LP-90-13 Zhisheng Huang — Logics for Belief Dependence
LP-90-14 Jeroen Groenendijk, Martin Stokhof — Two Theories of Dynamic Semantics
LP-90-15 Maarten de Rijke — The Modal Logic of Inequality
LP-90-16 Zhisheng Huang, Karen Kwast — Awareness, Negation and Logical Omniscience
LP-90-17 Paul Dekker — Existential Disclosure, Implicit Arguments in Dynamic Semantics

Mathematical Logic and Foundations

ML-90-01 Harold Schellinx — Isomorphisms and Non-Isomorphisms of Graph Models
ML-90-02 Jaap van Oosten — A Semantical Proof of De Jongh's Theorem
ML-90-03 Yde Venema — Relational Games
ML-90-04 Maarten de Rijke — Unary Interpretability Logic
ML-90-05 Domenico Zambella — Sequences with Simple Initial Segments
ML-90-06 Jaap van Oosten — Extension of Lifschitz' Realizability to Higher Order Arithmetic, and a Solution to a Problem of F. Richman
ML-90-07 Maarten de Rijke — A Note on the Interpretability Logic of Finitely Axiomatized Theories
ML-90-08 Harold Schellinx — Some Syntactical Observations on Linear Logic
ML-90-09 Dick de Jongh, Duccio Pianigiani — Solution of a Problem of David Guaspari
ML-90-10 Michiel van Lambalgen — Randomness in Set Theory
ML-90-11 Paul C. Gilmore — The Consistency of an Extended NaDSet

Computation and Complexity Theory

CT-90-01 John Tromp, Peter van Emde Boas — Associative Storage Modification Machines
CT-90-02 Sieger van Denneheuvel, Gerard R. Renardel de Lavalette — A Normal Form for PCSJ Expressions
CT-90-03 Ricard Gavaldà, Leen Torenvliet, Osamu Watanabe, José L. Balcázar — Generalized Kolmogorov Complexity in Relativized Separations
CT-90-04 Harry Buhrman, Edith Spaan, Leen Torenvliet — Bounded Reductions
CT-90-05 Sieger van Denneheuvel, Karen Kwast — Efficient Normalization of Database and Constraint Expressions
CT-90-06 Michiel Smid, Peter van Emde Boas — Dynamic Data Structures on Multiple Storage Media, a Tutorial
CT-90-07 Kees Doets — Greatest Fixed Points of Logic Programs
CT-90-08 Fred de Geus, Ernest Rotterdam, Sieger van Denneheuvel, Peter van Emde Boas — Physiological Modelling using RL
CT-90-09 Roel de Vrijer — Unique Normal Forms for Combinatory Logic with Parallel Conditional, a case study in conditional rewriting

Other Prepublications

X-90-01 A.S. Troelstra — Remarks on Intuitionism and the Philosophy of Mathematics, Revised Version
X-90-02 Maarten de Rijke — Some Chapters on Interpretability Logic
X-90-03 L.D. Beklemishev — On the Complexity of Arithmetical Interpretations of Modal Formulae
X-90-04 — Annual Report 1989
X-90-05 Valentin Shehtman — Derived Sets in Euclidean Spaces and Modal Logic
X-90-06 Valentin Goranko, Solomon Passy — Using the Universal Modality: Gains and Questions
X-90-07 V.Yu. Shavrukov — The Lindenbaum Fixed Point Algebra is Undecidable
X-90-08 L.D. Beklemishev — Provability Logics for Natural Turing Progressions of Arithmetical Theories
X-90-09 V.Yu. Shavrukov — On Rosser's Provability Predicate
X-90-10 Sieger van Denneheuvel, Peter van Emde Boas — An Overview of the Rule Language RL/1
X-90-11 Alessandra Carbone — Provable Fixed points in $I\Delta_0+\Omega_1$, revised version
X-90-12 Maarten de Rijke — Bi-Unary Interpretability Logic
X-90-13 K.N. Ignatiev — Dzhaparidze's Polymodal Logic: Arithmetical Completeness, Fixed Point Property, Craig's Property
X-90-14 L.A. Chagrova — Undecidable Problems in Correspondence Theory
X-90-15 A.S. Troelstra — Lectures on Linear Logic

1991

Logic, Semantics and Philosophy of Langauge

LP-91-01 Wiebe van der Hoek, Maarten de Rijke — Generalized Quantifiers and Modal Logic
LP-91-02 Frank Veltman — Defaults in Update Semantics
LP-91-03 Willem Groeneveld — Dynamic Semantics and Circular Propositions
LP-91-04 Makoto Kanazawa — The Lambek Calculus enriched with additional Connectives
LP-91-05 Zhisheng Huang, Peter van Emde Boas — The Schoenmakers Paradox: Its Solution in a Belief Dependence Framework
LP-91-06 Zhisheng Huang, Peter van Emde Boas — Belief Dependence, Revision and Persistence

Mathematical Logic and Foundations

ML-91-01 Yde Venema — Cylindric Modal Logic
ML-91-02 Alessandro Berarducci, Rineke Verbrugge — On the Metamathematics of Weak Theories
ML-91-03 Domenico Zambella — On the Proofs of Arithmetical Completeness for Interpretability Logic
ML-91-04 Raymond Hoofman, Harold Schellinx — Collapsing Graph Models by Preorders
ML-91-05 A.S. Troelstra — History of Constructivism in the Twentieth Century
ML-91-06 Inge Bethke — Finite Type Structures within Combinatory Algebras
ML-91-07 Yde Venema — Modal Derivation Rules
ML-91-08 Inge Bethke — Going Stable in Graph Models
ML-91-09 V.Yu. Shavrukov — A Note on the Diagonalizable Algebras of PA and ZF
ML-91-10 Maarten de Rijke, Yde Venema — Sahlqvist's Theorem for Boolean Algebras with Operators

Computation and Complexity Theory

CT-91-01 Ming Li, Paul M.B. Vitányi — Kolmogorov Complexity Arguments in Combinatorics
CT-91-02 Ming Li, John Tromp, Paul M.B. Vitányi — How to Share Concurrent Wait-Free Variables
CT-91-03 Ming Li, Paul M.B. Vitányi — Average Case Complexity under the Universal Distribution Equals Worst Case Complexity
CT-91-04 Sieger van Denneheuvel, Karen Kwast — Weak Equivalence
CT-91-05 Sieger van Denneheuvel, Karen Kwast — Weak Equivalence for Constraint Sets
CT-91-06 Edith Spaan — Census Techniques on Relativized Space Classes
CT-91-07 Karen L. Kwast — The Incomplete Database
CT-91-08 Kees Doets — Levationis Laus
CT-91-09 Ming Li, Paul M.B. Vitányi — Combinatorial Properties of Finite Sequences with high Kolmogorov Complexity
CT-91-10 John Tromp, Paul Vitányi — A Randomized Algorithm for Two-Process Wait-Free Test-and-Set
CT-91-11 Lane A. Hemachandra, Edith Spaan — Quasi-Injective Reductions

Computational Linguistics

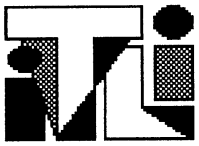
CL-91-01 J.C. Scholtes — Kohonen Feature Maps in Natural Language Processing
CL-91-02 J.C. Scholtes — Neural Nets and their Relevance for Information Retrieval

Other Prepublications

X-91-01 Alexander Chagrov, Michael Zakharyaschev — The Disjunction Property of Intermediate Propositional Logics
X-91-02 Alexander Chagrov, Michael Zakharyaschev — On the Undecidability of the Disjunction Property of Intermediate Propositional Logics
X-91-03 V. Yu. Shavrukov — Subalgebras of Diagonalizable Algebras of Theories containing Arithmetic
X-91-04 K.N. Ignatiev — Partial Conservativity and Modal Logics
X-91-05 Johan van Benthem — Temporal Logic
X-91-06 — Annual Report 1990
X-91-07 A.S. Troelstra — Lectures on Linear Logic, Errata and Supplement
X-91-08 Giorgie Dzhaparidze — Logic of Tolerance
X-91-09 L.D. Beklemishev — On Bimodal Provability Logics for Π_1-axiomatized Extensions of Arithmetical Theories
X-91-10 Michiel van Lambalgen — Independence, Randomness and the Axiom of Choice
X-91-11 Michael Zakharyaschev — Canonical Formulas for K4. Part I: Basic Results
X-91-12 Herman Hendriks — Flexibele Categoriale Syntaxis en Semantiek: de proefschriften van Frans Zwarts en Michael Moortgat
X-91-13 Max I. Kanovich — The Multiplicative Fragment of Linear Logic is NP-Complete
X-91-14 Max I. Kanovich — The Horn Fragment of Linear Logic is NP-Complete
X-91-15 V. Yu. Shavrukov — Subalgebras of Diagonalizable Algebras of Theories containing Arithmetic, revised version

The ITLI Prepublication Series

1986 86-01 The Institute of Language, Logic and Information
86-02 Peter van Emde Boas A Semantical Model for Integration and Modularization of Rules
86-03 Johan van Benthem Categorial Grammar and Lambda Calculus
86-04 Reinhard Muskens A Relational Formulation of the Theory of Types
86-05 Kenneth A. Bowen, Dick de Jongh Some Complete Logics for Branched Time, Part I Well-founded Time, Forward looking Operators
86-06 Johan van Benthem Logical Syntax
1987 87-01 Jeroen Groenendijk, Martin Stokhof Type shifting Rules and the Semantics of Interrogatives
87-02 Renate Bartsch Frame Representations and Discourse Representations
87-03 Jan Willem Klop, Roel de Vrijer Unique Normal Forms for Lambda Calculus with Surjective Pairing
87-04 Johan van Benthem Polyadic quantifiers
87-05 Víctor Sánchez Valencia Traditional Logicians and de Morgan's Example
87-06 Eleonore Oversteegen Temporal Adverbials in the Two Track Theory of Time
87-07 Johan van Benthem Categorial Grammar and Type Theory
87-08 Renate Bartsch The Construction of Properties under Perspectives
87-09 Herman Hendriks Type Change in Semantics: The Scope of Quantification and Coordination
1988 LP-88-01 Michiel van Lambalgen *Logic, Semantics and Philosophy of Language:* Algorithmic Information Theory
LP-88-02 Yde Venema Expressiveness and Completeness of an Interval Tense Logic
LP-88-03 Year Report 1987
LP-88-04 Reinhard Muskens Going partial in Montague Grammar
LP-88-05 Johan van Benthem Logical Constants across Varying Types
LP-88-06 Johan van Benthem Semantic Parallels in Natural Language and Computation
LP-88-07 Renate Bartsch Tenses, Aspects, and their Scopes in Discourse
LP-88-08 Jeroen Groenendijk, Martin Stokhof Context and Information in Dynamic Semantics
LP-88-09 Theo M.V. Janssen A mathematical model for the CAT framework of Eurotra
LP-88-10 Anneke Kleppe A Blissymbolics Translation Program
ML-88-01 Jaap van Oosten *Mathematical Logic and Foundations:* Lifschitz' Realizabiility
ML-88-02 M.D.G. Swaen The Arithmetical Fragment of Martin Löf's Type Theories with weak Σ-elimination
ML-88-03 Dick de Jongh, Frank Veltman Provability Logics for Relative Interpretability
ML-88-04 A.S. Troelstra On the Early History of Intuitionistic Logic
ML-88-05 A.S. Troelstra Remarks on Intuitionism and the Philosophy of Mathematics
CT-88-01 Ming Li, Paul M.B.Vitanyi *Computation and Complexity Theory:* Two Decades of Applied Kolmogorov Complexity
CT-88-02 Michiel H.M. Smid General Lower Bounds for the Partitioning of Range Trees
CT-88-03 Michiel H.M. Smid, Mark H. Overmars, Leen Torenvliet, Peter van Emde Boas Maintaining Multiple Representations of Dynamic Data Structures
CT-88-04 Dick de Jongh, Lex Hendriks, Gerard R. Renardel de Lavalette Computations in Fragments of Intuitionistic Propositional Logic
CT-88-05 Peter van Emde Boas Machine Models and Simulations (revised version)
CT-88-06 Michiel H.M. Smid A Data Structure for the Union-find Problem having good Single-Operation Complexity
CT-88-07 Johan van Benthem Time, Logic and Computation
CT-88-08 Michiel H.M. Smid, Mark H. Overmars, Leen Torenvliet, Peter van Emde Boas Multiple Representations of Dynamic Data Structures
CT-88-09 Theo M.V. Janssen Towards a Universal Parsing Algorithm for Functional Grammar
CT-88-10 Edith Spaan, Leen Torenvliet, Peter van Emde Boas Nondeterminism, Fairness and a Fundamental Analogy
CT-88-11 Sieger van Denneheuvel, Peter van Emde Boas Towards implementing RL
X-88-01 Marc Jumelet *Other prepublications:* On Solovay's Completeness Theorem
1989 LP-89-01 Johan van Benthem *Logic, Semantics and Philosophy of Language:* The Fine-Structure of Categorial Semantics
LP-89-02 Jeroen Groenendijk, Martin Stokhof Dynamic Predicate Logic, towards a compositional, non-representational semantics of discourse
LP-89-03 Yde Venema Two-dimensional Modal Logics for Relation Algebras and Temporal Logic of Intervals
LP-89-04 Johan van Benthem Language in Action
LP-89-05 Johan van Benthem Modal Logic as a Theory of Information
LP-89-06 Andreja Prijatelj Intensional Lambek Calculi: Theory and Application
LP-89-07 Heinrich Wansing The Adequacy Problem for Sequential Propositional Logic
LP-89-08 Víctor Sánchez Valencia Peirce's Propositional Logic: From Algebra to Graphs
LP-89-09 Zhisheng Huang Dependency of Belief in Distributed Systems
ML-89-01 Dick de Jongh, Albert Visser *Mathematical Logic and Foundations:* Explicit Fixed Points for Interpretability Logic
ML-89-02 Roel de Vrijer Extending the Lambda Calculus with Surjective Pairing is conservative
ML-89-03 Dick de Jongh, Franco Montagna Rosser Orderings and Free Variables
ML-89-04 Dick de Jongh, Marc Jumelet, Franco Montagna On the Proof of Solovay's Theorem
ML-89-05 Rineke Verbrugge Σ-completeness and Bounded Arithmetic
ML-89-06 Michiel van Lambalgen The Axiomatization of Randomness
ML-89-07 Dirk Roorda Elementary Inductive Definitions in HA: from Strictly Positive towards Monotone
ML-89-08 Dirk Roorda Investigations into Classical Linear Logic
ML-89-09 Alessandra Carbone Provable Fixed points in $I\Delta_0+\Omega_1$
CT-89-01 Michiel H.M. Smid *Computation and Complexity Theory:* Dynamic Deferred Data Structures
CT-89-02 Peter van Emde Boas Machine Models and Simulations
CT-89-03 Ming Li, Herman Neuféglise, Leen Torenvliet, Peter van Emde Boas On Space Efficient Simulations
CT-89-04 Harry Buhrman, Leen Torenvliet A Comparison of Reductions on Nondeterministic Space
CT-89-05 Pieter H. Hartel, Michiel H.M. Smid, Leen Torenvliet, Willem G. Vree A Parallel Functional Implementation of Range Queries
CT-89-06 H.W. Lenstra, Jr. Finding Isomorphisms between Finite Fields
CT-89-07 Ming Li, Paul M.B. Vitanyi A Theory of Learning Simple Concepts under Simple Distributions and Average Case Complexity for the Universal Distribution (Prel. Version)
CT-89-08 Harry Buhrman, Steven Homer, Leen Torenvliet Honest Reductions, Completeness and Nondeterminstic Complexity Classes
CT-89-09 Harry Buhrman, Edith Spaan, Leen Torenvliet On Adaptive Resource Bounded Computations
CT-89-10 Sieger van Denneheuvel The Rule Language RL/1
CT-89-11 Zhisheng Huang, Sieger van Denneheuvel, Peter van Emde Boas Towards Functional Classification of Recursive Query Processing
X-89-01 Marianne Kalsbeek *Other Prepublications:* An Orey Sentence for Predicative Arithmetic
X-89-02 G. Wagemakers New Foundations: a Survey of Quine's Set Theory
X-89-03 A.S. Troelstra Index of the Heyting Nachlass
X-89-04 Jeroen Groenendijk, Martin Stokhof Dynamic Montague Grammar, a first sketch
X-89-05 Maarten de Rijke The Modal Theory of Inequality
X-89-06 Peter van Emde Boas Een Relationele Semantiek voor Conceptueel Modelleren: Het RL-project
1990 *Logic, Semantics and Philosophy of Language*
LP-90-01 Jaap van der Does A Generalized Quantifier Logic for Naked Infinitives
LP-90-02 Jeroen Groenendijk, Martin Stokhof Dynamic Montague Grammar
LP-90-03 Renate Bartsch Concept Formation and Concept Composition
LP-90-04 Aarne Ranta Intuitionistic Categorial Grammar
LP-90-05 Patrick Blackburn Nominal Tense Logic
LP-90-06 Gennaro Chierchia The Variablity of Impersonal Subjects
LP-90-07 Gennaro Chierchia Anaphora and Dynamic Logic
LP-90-08 Herman Hendriks Flexible Montague Grammar
LP-90-09 Paul Dekker The Scope of Negation in Discourse, towards a flexible dynamic Montague grammar
LP-90-10 Theo M.V. Janssen Models for Discourse Markers
LP-90-11 Johan van Benthem General Dynamics
LP-90-12 Serge Lapierre A Functional Partial Semantics for Intensional Logic



Instituut voor Taal, Logica en Informatie
Institute for Language, Logic and Information

Faculteit der Wiskunde en Informatica
(Department of Mathematics and Computer Science)
Plantage Muidergracht 24
1018TV Amsterdam

Faculteit der Wijsbegeerte
(Department of Philosophy)
Nieuwe Doelenstraat 15
1012CP Amsterdam

KOHONEN FEATURE MAPS IN NATURAL LANGUAGE PROCESSING

J.C. Scholtes
Department of Compuational Linguistics, Faculty of Arts
University of Amsterdam
email: scholtes@alf.let.uva.nl
mail: Dufaystr.1, 1075GR Amsterdam
fax: +31 20 6710793

ITLI Prepublications
for Computational Linguistics

Received October 1991

This research is supported by
MSC Beheer BV, Amsterdam

Abstract

In the 1980s, backpropagation (BP) started the connectionist bandwagon in Natural Language Processing (NLP). Although initial results were good, some critical notes must be made about the blind application of BP. Most such systems require that contextual and semantical features are added manually by structuring the input set. Moreover, these models form a small approximation of the brain structures known from neural sciences. They do not adapt smoothly to a changing environment and can only learn input/output pairs. Although these disadvantages of the backpropagation algorithm are commonly known and accepted, other more plausible learning algorithms, such as unsupervised learning techniques, are still rare in the field of NLP. The main reason is the high complexity of unsupervised learning methods when applied in the already complex field of NLP. However, recent efforts implementing unsupervised language learning have been made, resulting in interesting conclusions.

Taking off from this earlier work, this paper presents a recurrent self-organizing model (based on an extension of the Kohonen feature map), which is capable of deriving contextual (and some semantical) information from scratch. The model implements a first step towards an overall unsupervised language learning system. Simple linguistic tasks such as single word clustering (representation on the map), syntactical group formation, derivation of contextual structures, string prediction, grammatical correctness checking, word sense disambiguation and structure assigning are carried out in a number of experiments. The performance of the model is as least as good as achieved in recurrent backpropagation, and at some points even better (e.g. unsupervised derivation of word classes and syntactical structures). Although premature, the first results are promising and show possibilities for other even more biologically-inspired language processing techniques such as real Hebbian, Genetic and Darwinistic models. Further research must overcome limitations still present in the extended Kohonen model, such as the absence of within layer learning, restricted recurrence, no look-ahead functions (absence of distributed or unsupervised buffering mechanisms) and a limited support for an increased number of layers.

Background

The importance of connectionism in Natural Language Processing (NLP) has been advocated by many researchers lately. The ability to represent knowledge in a distributed way without the addition of explicit knowledge structures and loss of generality convinced many in the field of the usability of such techniques in NLP [Graubard, 1988]. However, most neurally inspired models implement only a fraction of the knowledge found in neural sciences. The main reason for this is the tremendous complexity of biological neural nets. Beside the large amount of neurons and connections needed, there is the mathematical complexity of the feed forward and learning rules.

The popular backpropagation algorithm is a very restricted example of such neurally inspired networks. There are no connections within a layer, only between layers. So, only connections between these (usually three) layers learn. Furthermore, a small number of architectures feature recurrent fibres. If one puts the biological neural nets next to these models, just a small shadow of resemblance remains. What is known of neural structures in the cortex tells us at least that there are connections within a layer, which are learned just as the inter-layer neurons are. Brain structures consist of multiple layers. Next, there are many recurrent fibres connecting neurons at different layers. All connections learn due to synaptical plasticity, there is constant adaption to a changing environment by reformations on the cortex map. The model converges to something else than a zero-point or zero-line in a multidimensional space: more realistic models do not converge in the way proposed in various models, but show chaotic behaviour.

Backpropagation is a so-called *supervised* learning algorithm where an external teacher adjusts the weights [McClelland et al., 1986a] [McClelland et al., 1986b] [Rumelhart et al., 1986]. *Unsupervised* learning rules (without an external teacher) have been developed in different directions. All of them are based on a competitive learning principle: *if a pattern fits best on a certain region on the map, adjust the weights so it fits even better on this position the next time it is presented* [Rumelhart et al., 1985]. These rules in their turn are all derived from the Hebbian learning rule: *if two neurons are activated at the same time, increase the connection strength between them.* Globally we observe three main streams of research efforts. First, there are the competitive algorithms of Grossberg and Kohonen. Grossberg uses a two-layer, fully interconnected model based on competitive learning principles: *Adaptive Resonance Theory ART* [Grossberg, 1980] [Grossberg, 1988] [Carpenter et al., 1988]. Kohonen developed a one-layer map where all input sensors are connected to all neurons. The map doesn't fire but results in the formation of a topological map of the input sensor values: a *Self-Organizing Map.* Next, the more Hebbian models of Von der Malsberg and Linsker can be distinguished [Hebb, 1949], [Malsberg, 1973], [Linsker, 1988]. Here, the learning rule adapts all connections in all directions. Finally, recent developments indicate good possibilities for evolutionary models based on *Darwin's* selection theories and the so-called *Genetic Algorithms.* These models implement population theories in neural nets: successful populations multiply faster than ones that are unsuccessful (with respect to some quality measurement) [Holland, 1975] [Goldberg et al., 1988], [Goldberg, 1988]. The unsupervised learning algorithms mentioned above are ordered in increasing complexity.

Unsupervised learning might be defined as the total absence of a central control mechanism which implements an external teaching unit. There are different interpretations of this definition. One can abandon a central control mechanism totally or accept it at the neuronal group level. There is no evidence for central control mechanisms in the human brain. Neural sciences indicate a locally distributed organization. Specific functions are implemented in specific parts of the brain. Moreover, locally this knowledge is distributed implementing association, generalization and adaptation mechanisms. A low level normalization process can be seen as a necessary locally specific function, implementing the overall unsupervised learning process. But, it could also be abandoned completely on grounds of not being unsupervised (some central mechanism must control the normalization). The neurological plausibility of this decision has a very vague border: what is local and what is locally distributed? At first sight central control mechanisms must be avoided. All knowledge should be distributed. But if a subnet implements a specific function, different learning rules and inter-layer connections can be seen as a locally specific functions making that part of the net especially suitable to implement a certain function.

In certain cases the application of neurally inspired methods in NLP is obvious. In others, it can be real hard to develop them. Spelling corrections, lexical access, word sense disambiguation and generalization are implemented by relatively simple means. These problems use the most basic and implicit present characteristic features of neural nets: association and generalization caused by the parallel distributed knowledge representation. On the other hand, to solve the representation of time (or sequences needed to define grammatical correctness and to carry out a sentence parse) in parallel systems is quite a problem. In sequential processing, contextual information of sequential strings is gotten for free. However, in parallel processing one can either add explicit time marks (increasing the dimension of the input vectors), or concatenate different parts of the input towards one large vector (the window principle), or add buffers to the system, or use recurrent fibres. Where the first two options are not really expected to be found in biological systems, the second one has its roots in the *seven plus/minus two* window theory of human short-term memory [Miller, 1957]. But, there is a psychological problem, windows as implemented in current neural nets use seven plus/minus two characters, phonemes or words as input. Results from psychology indicate that the size of a short-term memory is about seven plus/minus two, but the question is seven plus/minus *what*. Different people have different objects in short-term memory. Therefore a simple window mechanism on character, phoneme or word level doesn't suffice. The third option has not yet been worked out in relation to neural nets (at least, not without the addition of an overall control mechanism). A buffering system working at different hierarchical levels can overcome the disadvantages of window systems. However, results are juvenile [Powers, 1989]. The use of recurrent fibres already had great impact in the connectionist language processing community [Elman, 1988] [Cleeremans et al., 1989] [Allen, 1990]. Several features of recurrent fibres are analysed thoughtfully, making them suitable in an unsupervised environment. Fibres as added here result in a model able to recognize finite state grammars (FSG). However, finite state machines (FSM) alone are too restricted for complete natural language processing. The occurrence of an object in a FSG string depends, just like in many NLP sentences, only on objects encountered in the near past. One needs more powerful mechanisms, such as the ability to anticipate on forthcoming string elements, to implement some form of context sensitivity.

Other important questions in unsupervised (connectionist) language learning concern the need for semantical additions in the learning set, the importance of negative information and the need for recursion. In other words, how much can be learned without semantics.

From a theoretical point of view the implementation of recursion in neural nets is very interesting. However, do we really need recursion to define grammatical structures or can natural language be recalled by using association in a distributed neural net? The research carried out tries to provide an answer to the questions posed.

After the evaluation of backpropagating connectionist NLP systems [Scholtes, 1990], our current project aims at the application of unsupervised learning mechanisms to NLP problems. The present paper decribes results obtained with an extended Kohonen model. The model performs a number of linguistic tasks. As mentioned, the author is aware of the implicit restrictions made in the self-organizing map, but by investigating a recurrent Kohonen map, the research serves two purposes. First, it will demonstrate that the Kohonen map can be very useful in NLP and other symbolic processing jobs [Hemani et al., 1990], although so far it is mainly used in vector quantization processes and not known for its symbolic processing abilities yet [Rubner et al., 1990]. Second, the model learns linguistic structures from unformatted strings passing by: an unsupervised learning process applied to NLP. Future research may use other unsupervised learning algorithms in NLP, based on experiences obtained with the relatively simple Kohonen feature map.

This research is part of a study towards the usability of connectionist learning methods in natural language processing. This in its turn is part of a long term project developing new methods in computational-linguistic areas such as data-oriented parsing, early language acquisition, and structural/semantical disambiguation.

Introduction

The Kohonen formalism is a competitive learning algorithm [Kohonen, 1982a, 1982b, 1982c, 1984, 1988, 1990a, 1990b]. A two-dimensional map is constructed in a rectangular or hexagonal structure from individual neurons. Each neuron has a number of input sensors with an input activation and an input weight. All neurons have the same number of input sensors. The learning rule acts in the following way. First, copy the activation values of an input element into all input activation sensors of all neurons. Next, determine the best match by finding the neuron with the minimum mathematical (e.g. euclidean) distance between input and weight values. Then, adapt the weights of the neurons within a certain region of this minimum, so they'll recognize the current input vector better in the near future. After numerous cycles, a topological map is formed, holding related elements in neighbouring regions.

Obvious applications of Kohonen feature maps in language processing can be found in [Miikkulainen et al., 1988a, 1988b], and [Schyns, 1990a, 1990b]. Although actual learning is done with a supervised learning method, the Kohonen formalism plays a conceptually significant role. Another attempt to use Kohonen feature maps in NLP can be found in [Ritter et al., 1989b, 1990]. Here, words are taught to a single Kohonen map by feeding the concatenation of a symbol code and a context code into the input sensors, resulting in a so called *semantotopic* map.

To achieve automatic derivation of syntactic features as well as syntactic structures, one has to use a method similar to the one as proposed by Ritter: add implicit context sensitivity to the system. There are several methods to do this. First there is the window principle, as used by Ritter, which results in a restricted sentence length. In [Kohonen et al., 1981] the authors propose a centrally guided buffering mechanism to implement temporal processing abilities. Although this is a practical solution, we prefer a more distributed and unsupervised mechanism. In [Tavan et al., 1990] sensor values are exchanged between a sensor and a feature map, resulting in the formation of an associative memory. Although interesting, this is unsuitable for our current purpose. Next, [Thacker et al., 1990] describes the design of an unsupervised multi-layer context-sensitive model, which uses recurrent fibres. This is a problem in the Kohonen learning algorithm: it lacks a notion of firing. [Kangas, 1990] provides a solution for this problem, making it possible to use recurrent fibres in a multi-layer Kohonen map.

Kangas calculates the degree of correspondence between input values and weight values for all neurons on the map. Every neuron is represented by a dimension of a vector. This vector expresses the activation of the feature map to an input vector. By averaging this vector in time, the system gets more or less sensitive to changes and noisy input. The result of this vector is fed back into the first layer as contextual information. So, the input vector of the first layer consists of the concatenation of a (randomly assigned) symbolic part and a recurrent contextual part. The output vector of the first map serves as input for the second map.

Over time, the dimension of the input vectors of the second map definitely gets too large for efficient simulations. Therefore, it is normalised and reduced in dimension. Although normalization and dimension reduction are supervised processes, they can also be interpreted as a (natural) resource usage process (if one neuron uses more chemical resource to obtain a voltage increase, there is less left for other neurons, resulting in a voltage decrease) [Malsberg, 1973]. On the other hand, this process should not be necessary if enough computational power would be available. By learning both maps according to the Kohonen formalism, the first map forms an ordering with syntactically equivalent words in subsequent regions, and the second map holds related contextual structures in neighbouring regions. All resulting from single strings just passing by.

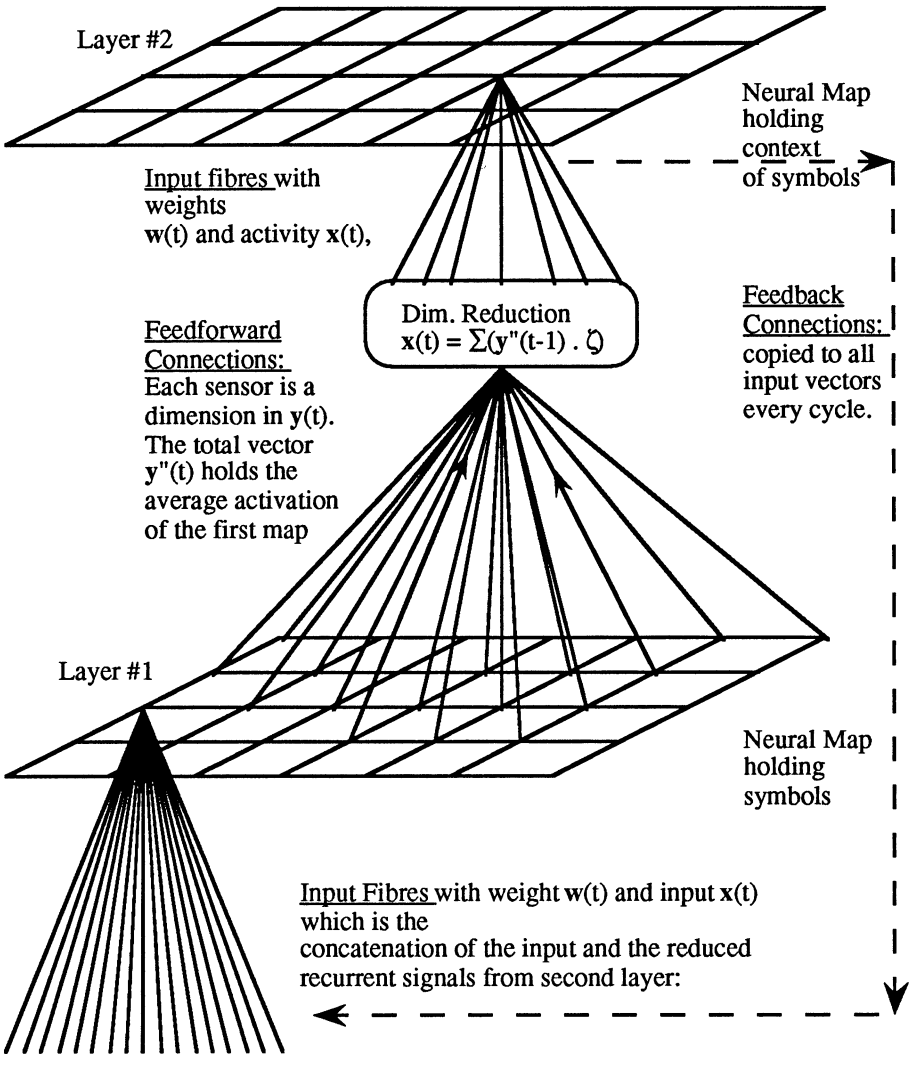
The presented model only exhibits left-context sensitivity. However, Natural Language Processing needs something more powerful. Often, the information that has been processed in the near past provides enough context to disambiguate what follows. But sometimes one needs information from the more remote past as well as the future in order to disambiguate complex

grammatical structures. The model shall not succeed in this without the addition of a buffering mechanism (or memory), capable of processing information in a way which is common knowledge in sequential processing. The main problem is the lack of understanding of a possible buffering system which is not controlled by a supervised mechanism, which would result in a bottle-neck or in an unrealistic model of human information processing. Future research has to clarify this shortcoming of the model as it is proposed here.

In order to be useful in sentence processing systems, the neural net should at least meet the following objectives:

- Cluster (equal) words in regions of the map
- Word class derivation on syntactical and semantical grounds
- Predict next elements in a string
- Determine grammatical correctness of a string (accept or reject it)
- Disambiguate word senses
- Attach a structure to a sentence

By combining the neural net with a conventional shell, which provides information to be processed in a convenient way, a sentence processing model is simulated where the model learns everything it knows from an unsupervised learning algorithm.



Definitions

$N^{(1)}$	Number of Neurons in first layer
$N^{(2)}$	Number of Neurons in second layer
n_S	Dimension of Symbol vector
n_r	Dimension of Recurrent vector
n	Dimension of Input vector
L	Layer number
$M^{(L)}$	Set of all neurons in map L
k,v	Number of neuron on map
$x^{(1)}{}_S(t)$	Symbol Input of the first layer
$w^{(1)}{}_S(t)$	Symbol Weights of the first layer
$x^{(1)}{}_r(t)$	Recurrent Input of the first layer
$w^{(1)}{}_r(t)$	Recurrent Weights of the first layer
$x^{(1)}(t)$	Concatenated Input of the first layer
$w^{(1)}(t)$	Concatenated Weights of the first layer
$y^{(1)}(t)$	Activation Measure of the first layer
$y^{(1)'}(t)$	Sharpened Activation Measure of the first layer
$y^{(1)''}(t)$	Averaged Activation Measure of the first layer
ζ_i	Vector representing dimension i of $N^{(1)}$
$x^{(2)}(t)$	Input of the second layer
$w^{(2)}(t)$	Weights of the second layer

$Dim(x^{(1)}{}_S(t))$	=	n_S		
$Dim(x^{(1)}{}_r(t))$	=	n_r		
$Dim(x^{(1)}(t))$	=	n	=	$n_S + n_r$
$Dim(y^{(1)}(t))$	=	$N^{(1)}$		
$Dim(\zeta_i)$	=	n_r		
$Dim(x^{(2)}(t))$	=	n_r		
$Dim(w^{(2)}(t))$	=	n_r		

Algorithm

The learning algorithm consists globally of 8 steps. Below, they are discussed in detail. The number between square brackets corresponds with the step of the algorithm in the box on page 10.

First, a number of random vectors is generated to represent codes for the symbols encountered in the learn set. These vectors have a dimension equal to the amount of input sensors of the first map. The input vector $x^{(1)}(t)$ and the weight vector $w^{(1)}(t)$ of the first layer are concatenations of the vector code for an element $x^{(1)}{}_S(t)$ and the recurrent context $x^{(1)}{}_r(t)$, respectively of their weights $w^{(1)}{}_S(t)$ and $w^{(1)}{}_r(t)$. Random vectors are substituted for $x^{(1)}{}_S(t)$. Because the dimension of an output vector of the first map is reduced for reasons of complexity only, a second set of random vectors is generated, each representing a dimension of the recurrent input vector $x^{(1)}{}_r(t)$. These vectors form a random basis for the recurrent input space

and are indicated by the symbol ζ_i, where i is the dimension number. Later on, an exact definition of this dimension reduction shall be given [step 1].

Next, an input sentence from the learn set is split into separate objects, each representing a word. An external algorithm determines unique elements occurring in the learn set and assigns (at random) a code from step 1 to each of these words [step 2].

Depending on the number of learn cycles, the model selects sentences from the learn set at random. Its separate words are successively fed into the system one by one. This step is repeated for the desired number of learn cycles [step 3].

Steps 4 to 8 are repeated for every word in the sentence. As stated, the input vector is a concatenation of a symbolic and a recurrent vector. The symbol vector is one of the random vectors generated in step 1. If the first word is fed into the system, there is no context available. Therefore the recurrent vector equals the zero-vector. Then, the input vector is the concatenation of the symbol vector and the zero-vector. If, on the other hand, there has been previous input, the input vector is the concatenation of the symbol and recurrent vector. Weights are always a concatenation of the symbolic and recurrent weights, otherwise previously learned information would be ignored. The result of the input concatenation is fed into the first layer of the model [step 4].

$y^{(1)}(t)$ represents the activation measure of the first layer. If a word is the starting element of a string, the previous activation of the first layer equals the zero-vector (this is the activation of the former element in a string, used to average the input). The activation value of a neuron on the first map is always calculated by subtracting the input of neuron i on the map from the weight of that specific neuron. The smaller this result, the better the neuron represents the input vector. This value is increased by a small value δ (to avoid dividing by zero), and inverted. So high values correspond to perfect maps. The same calculation is repeated for every neuron on the first map, resulting in a vector with as many dimensions as neurons on the map. As a matter of fact, every dimension represents the measure of correspondence between the input vector and a neuron on the map.
To avoid arithmetic influences of the random codes generated, this vector is normalized towards $y^{(1)'}(t)$ by dividing its square value with the summation of the square values of all neurons on the map (which step can be repeated several times). As a result, the summation of all the elements of the output vector equals one. To avoid the system from being too sensitive to changes, $y^{(1)''}(t)$, the averaged activation in time is determined by adding the activation at *t-1* and the activation at *t*, multiplying the two elements with a value ω, and $1-\omega$ respectively: the memory rate of the system. A large ω results in short memory and sensitivity to changes. A small value causes the system to adapt slowly to changes in the input [step 5].

A number of vectors ζ_i was generated randomly in step [1], one for each dimension of $y^{(1)''}(t)$. The vector copied into the input of the second layer: $x^{(2)}(t)$, and into the recurrent fibres of the first: $w^{(2)}(t)$, is determined by summarizing the multiplications of the separate dimensions of $y^{(1)'}(t)$ with the corresponding vector in ζ_i. The legitimacy of this dimension reduction is based on the heuristic that most of the elements in $y^{(1)''}(t)$ are about zero and their norm equals 1. Therefore, this operation conserves the main characteristics of the original vector. The number of fibres is reduced enormously, caused by the reduction of the vector dimension with a factor 10. So, even large maps learn complex representations within reasonable time limits. See [Ritter et al., 1989] for a proof of the legitimacy of this operation [step 6].

Till now, the only task performed has been the feed forward of the input vector through the network. If the word in question is the first word of a sentence, it is combined with a zero-vector (representing the situation in which no context is known) and the recurrent vector equals the zero-vector. Because all string starting words have the same recurrent part, they shall form neighbouring regions on the map. On the other hand, if the word is not the preceding one in a string, the recurrent vector is determined by the activation measurement of the first map caused by the previous word. The map organizes sequencing words in the same region, based on equality of the recurrent vectors (although words on the first map keep moving until a perfect self-organization is formed, within a certain time interval, positions are quite stable). This delicate balance between the recurrent and symbolic vector results in the organization desired. The algorithm learns the first layer as well as the second one. The first with vector concatenations from step 1, the second with the input vector of step 6 and the weight vector of the second layer. Therefore, the model applies the Kohonen learning rule. First, we determine the best match for this element on the map by calculating the minimum euclidean distance between the input vector and the weight vector for all neurons on the map. This minimum holds for the neuron which represents the input vector best [step 7].

If the weights of this neuron and the surrounding ones are adopted in this way, they represent the input vector better next time it occurs. Two reasons guarantee convergence towards a self-organizing state (at least, if the number of learn cycles is large enough). In the first place, the learning rate (the measure by which weights adopt to new values) and the region size (the number of neurons in the direct neighbourhood which are adopted) decrease as a bell shaped function in time. Both variables start with a large value which decreases slowly towards zero. Therefore the changes in the weights reaches zero as the number of learning cycles increases towards infinity. Secondly, if a self-organizing state exists, it shall be reached in time if the input is presented randomly (although there is no direct analogy, this effect can be compared with a characteristic of hidden markov chains: if there exists an absorbing state and one walks randomly from state to state in the markov model, then the absorbing state *shall* be reached in time. Similarly, there is no escape from self-organization). Furthermore, the larger the physical distance between a neuron and the optimum position on the map, the smaller the adjusting of the weights of this neuron shall be [step 8].

One can expect convergence to take place on two grounds. First , the first map organizes on properties of the symbolic part of the input vector. Equal words shall be ordered in uniform classes. Second, there is the organization triggered by the recurrent fibres, resulting in regions of word classes which are used in the same context -- not exactly syntactic classes, but more something like substitutionally or semantically equal words. Both organizations influence each other, resulting in chaotic behaviour of map formations. The second map follows the organization on the first map, and shall therefore only start to get organized as the first map has reached some initial state of self-organization. Convergence times shall be quite long, considering the nature of two such self-organizing processes. Furthermore, because the symbolic and recurrent vectors have the same norm, neither of them can turbo-charge the convergence process. One has to await the moment where every element is on its correct position according to the symbolic as well as the contextual constraints. Only then, convergence towards the self-organizing state occurs.

Activation and Learning Algorithm

1. *Generate random codes for the symbol representation:* $x^{(1)}{}_S$ *and the dimension reduction vectors:* ζ_i.
2. *Split input string in separate parts, each part holding exactly one symbol.*
3. *Feed all the single symbols one by one into the net by assigning their random codes to the sensor activation values. Repeat step 4 to 8 for all elements of the learn set.*
4. *Set input values on the sensors of the first layer and concatenate the symbol and recurrent vectors:*

$$\mathbf{x}^{(1)}(t) = \begin{cases} [\mathbf{x}^{(1)}{}_S(t) + \underline{\mathbf{o}}] & \text{if first element string} \\ [\mathbf{x}^{(1)}{}_S(t) + \mathbf{x}^{(1)}{}_r(t)] & \text{else} \end{cases}$$

$$\mathbf{w}^{(1)}(t) = [\mathbf{w}^{(1)}{}_S(t) + \mathbf{w}^{(1)}{}_r(t)]$$

5. *Calculate the average activation of the first map:*

$$y^{(1)}(t) = 1.0 / ((w^{(1)}(t) - x^{(1)}(t))^2 + \delta)$$

$$Y(t) = \Sigma_{i=1,N^{(1)}} (y_i^{(1)}(t))^2$$

$$y^{(1)'}(t-1) = \underline{o} \quad \text{if first element string}$$

$$y^{(1)'}(t) = (y^{(1)}(t))^2 / Y(t)$$

$$y^{(1)''}(t) = (\omega \cdot y^{(1)'}(t) + (1 - \omega) \cdot y^{(1)'}(t - 1)) / \omega$$

6. *Reduce dimension* $y^{(1)''}(t)$ *from* $N^{(1)}$ *to* n_r. *Copy result into the activation sensors of the second map and into the recurrent fibres of the first map:*

$$\mathbf{x}^{(2)}(t) = \Sigma_{i=1,N^{(1)}} (y_i^{(1)''}(t) \cdot \zeta_i)$$

$$\mathbf{x}^{(1)}{}_r(t+1) = \mathbf{x}^{(2)}(t)$$

7. *Determine minimum map L: neuron v. This neuron has the net's best match between the input values and its weight values:*

$$v : \forall k \; \|\mathbf{w}_v(t) - \mathbf{x}(t)\| \le \|\mathbf{w}_k(t) - \mathbf{x}(t)\|$$

for L = 1,2 and k element of $M^{(L)}$

8. *Update all weights in the map according to the Kohonen learning rule:*

$$\mathbf{w}^{(L)}{}_k(t+1) = \mathbf{w}_k{}^{(L)}(t) + \varepsilon(t) \cdot \Phi_{kv} \cdot (\mathbf{x}^{(L)}(t) - \mathbf{w}_k{}^{(L)}(t)), \; L = 1,2$$

$$\Phi_{rs} = e^{-(\| k - v \| \; 2\sigma(t)^2)}$$

$$\varepsilon(t) = \varepsilon_{max} \cdot (\varepsilon_{min}/\varepsilon_{max})^{t/t_{max}}$$

$$\sigma(t) = \sigma_{max} \cdot (\sigma_{min}/\sigma_{max})^{t/t_{max}}$$

where:

ω	$\in$	[0,1]	Memory Rate
ε_{max}	$\in$	[0,1]	Start Learning Rate
ε_{min}	$\in$	[0,1]	Final Learning Rate
σ_{max}	=	$\sqrt{(N^{(L)})} / 2$	Start Region Size
σ_{min}	$\in$	[0,1]	Final Region Size
$\| k - v \|$	=	Physical distance on map from neuron k to v	

Experimental Results

All simulations were implemented on a Sun Sparc Station IPC in C. The simulator used sentences of three different input types to evaluate the model: sentence of the form {NOUN VERB NOUN}, sentences of the form {John Loves Mary} [Elman, 1988], and strings of the form {ABC} [Cleeremans et al., 1989].

Different Input Types

To start, there are 16 simple sentences of syntactic constituents (see table: *Type 1: Syntactic Sentence Structures*.). This formalism was used to illustrate the formation of a topological map of syntactic equivalents. Then, normal words were substituted for the constituents, resulting in the final learn sentences (see *Word Class Substitutes* and *Type 2: Some of the Sentences Generated ... Substitutes*). These sentences show how the model derives syntactic classes from flat strings. Because both input types provide no insight in the grammatical capabilities of the model, a third type (generated by a finite state machine) was introduced to evaluate the model's grammatical learning power.

EXAMPLES {NOUN-VERB-NOUN}	
0	NOUN-HUM VERB-EAT NOUN-FOOD
1	NOUN-HUM VERB-PERCEPT NOUN-INANIM
2	NOUN-HUM VERB-DESTROY NOUN-FRAG
3	NOUN-HUM VERB-INTRAN
4	NOUN-HUM VERB-TRAN NOUN-HUM
5	NOUN-HUM VERB-AGPAT NOUN-INANIM
6	NOUN-HUM VERB-AGPAT
7	NOUN-ANIM VERB-EAT NOUN-FOOD
8	NOUN-ANIM VERB-TRAN NOUN-ANIM
9	NOUN-ANIM VERB-AGPAT NOUN-INANIM
10	NOUN-ANIM VERB-AGPAT
11	NOUN-INANIM VERB-AGPAT
12	NOUN-AGGRESS VERB-DESTROY NOUN-FRAG
13	NOUN-AGGRESS VERB-EAT NOUN-HUM
14	NOUN-AGGRESS VERB-EAT NOUN-ANIM
15	NOUN-AGGRESS VERB-EAT NOUN-FOOD

Type 1: Syntactic Sentence Structures

0	NOUN-HUM	man, woman
1	NOUN-ANIM	cat, mouse
2	NOUN-INANIM	book, rock
3	NOUN-AGRESS	dragon, monster
4	NOUN-FRAG	glass, plate
5	NOUN-FOOD	cookie, bread
6	VERB-INTRAN	think, sleep
7	VERB-TRAN	see, chase
8	VERB-AGPAT	move, break
9	VERB-PERCEPT	smell, see
10	VERB-DESTROY	break, smash
11	VERB-EAT	eat

Word Class Substitutes

In the first experiment sentences from type 1 were taught to the net. The examples shown above are the only ones learned (in random order, multiple times). The seond experiment used examples of type 2, which were generated from type 1 syntactical structures and word class substitutes. The algorithm randomly selects words for each class and substitutes it in the syntactical framework. Although simple, the sentences produced are complicated enough to demonstrate the self-organizing capabilities of the model.

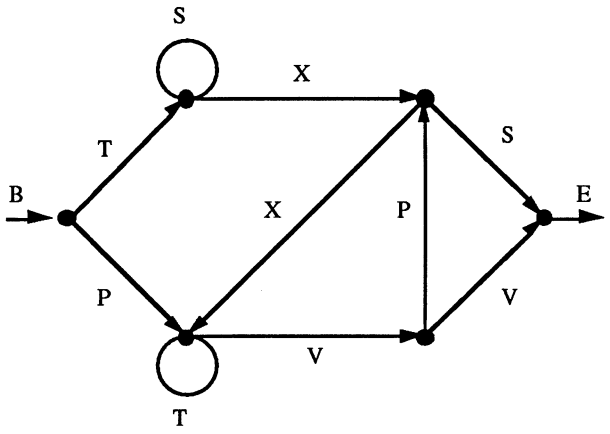
The result shown below is a subset of the 100.000 sentences generated. The first number indicates the number selected from 100.000 sentences, the second number indicates the learn cycle number (these examples are captures from a learn session).

```
        EXAMPLES {JOHN LOVES MARY}

4303     0: Jim works seldom
2539     1: Jim eats beer
4471     2: Jim walks slowly
3791     3: dog drinks water
4013     4: Mary hates bread
2998     5: Bob drinks seldom
4320     6: Bob runs well
2980     7: cat walks seldom
1664     8: Jim eats water
3407     9: Jim likes cat
 161    10: Bob hates water
2698    11: Mary hates water
  83    12: horse runs poorly
3528    13: cat likes bread
```

```
4115    24: cat likes cat
4037    25: dog eats water
3329    26: Jim works slowly
2645    27: Jim sells slowly
 992    28: dog eats seldom
 585    29: Bob speaks slowly
1198    30: cat eats slowly
2337    31: Jim speaks poorly
 563    32: Mary hates bread
4865    33: Mary drinks seldom
4764    34: cat eats seldom
 292    35: Mary hates dog
 240    36: cat walks well
1065    37: dog eats water
```

Type 2: Some Sentences Generated from Syntactic Structures and Words Class Substitutes



Finite State Machine (FSM)

```
                EXAMPLES {ABC}

  No    Cycle    String

 2581      0:   bpttvve
  692      1:   btxxvve
 5285      2:   bpvpse
  676      3:   btxxtvve
 5387      4:   btststsptvbtxxvve
 7592      5:   bpvpse
 9899      6:   bpvpse
 4372      7:   btxse
 5525      8:   bpttvve
 3281      9:   bptvpse
 2086     10:   bpvve
 7158     11:   bpttvpse
 7117     12:   bpvve
 9719     13:   bpvpse
  176     14:   bpvpxtvpsebtsptvve
 8374     15:   bpvpse
 8611     16:   bpvpxtvpxtbpvve
   15     17:   btxxttvve
  793     18:   bptvve
 6454     19:   btspvve
```

Type 3: Strings Generated From the FSM

Type 3 sentences are used to test the systems ability to predict and recognize elements of strings generated by a Finite State Machine (FSM) [Cleeremans et al., 1989]. A FSM generates strings as shown above in: *Finite State Machine.* If a state results in two directions (such as the first state, where one can choose between a *T* and a *P*), one is chosen randomly with a probability distribution of 0.5. The strings shown form a subset of more than 1 million generated (See: *Type 3: Strings Generated From the FSM*). The FSM has some characteristics making it interesting for simulations. First, there are multiple choices in every state. Secondly, the recurrent transitions with *S* and *T* can result in long sequences, testing the network's

memory capacity. Third, all elements occur at different transition places, forcing the system to remember (and use) a lot of context to determine grammatical correctness or to predict the next element in a string.

Simulation Parameters and Convergence

The parameter values used for the simulations were about the same for all three different types of input (variables specified after step 8 in the algorithm). However, different parameters were mainly influencing (temporal and element) memory capacity and convergence speed. The following model constants worked best: $\omega = 0.3333$, $\varepsilon_{max} = 0.80$, $\varepsilon_{min} = 0.05$, $\sigma_{max} = 4.5$, $\sigma_{min} = 0.5$.

After some initial simulations, the relation between the amount and norm of input– and recurrent fibres seemed very important. If the norm of the symbolic part was larger than the recurrent one, an ordering based on symbolic grounds instead of contextual ones developed. If, on the other hand, the recurrent norm was larger than the symbolic, the entire map converged towards the first element encountered in the learning sequence. Therefore it is very important that the norm as well as the number of fibres of the symbolic vector and the recurrent vector are equal. In this context, the dimension reduction of the input vector in the second map plays an important role, based on other than complexity reasons. Without this reduction, the norm of the recurrent fibres would be too small, resulting in an ordering based on the internal coding of the symbolic elements. Now, both vectors are constructed from the same random set.
The convergence process seemed to be quite complex. This is understood if one realises that there are in fact two non-linear processes influencing each other.

At first sight, convergence seemed based on good luck rather than on logical foundations. However, a number of parameters can influence the convergence process. First, there is the sharpening of $y^{(1)'}(t)$. If this vector is normalised multiple times by dividing its square value with $Y(t)$, it represents the most activated neuron better and better, resulting in a faster convergence of the first map. Secondly, a large ω means shorter memory, but a faster convergence with small sentences. If strings become larger, ω must definitely be decreased to avoid the system from having only single-word left-context sensitivity. A reasonable heuristic for the value of ω is 1/(average sentence length). Furthermore, there are the values for ε and σ, which have their specific influence on the self-organizing process as described in [Ritter et al., 1988]. Although convergence can be guided, convergence times are large and show chaotic behaviour, resulting in a process which is complex and hard to monitor. Last but not least, one must be aware of the fact that the organization of the second map (contextual structures) only starts after the first map is about to be organized. This implies that multi-level organization in models similar to the one proposed here, takes probably at least twice as long as in single layer models.

The Internal Coding of the Symbols

The internal coding of the symbols was generated and assigned randomly, although the internal distance was constant (e.g. only 0.00, 0.33, 0.66 and 1.00 were used as legitimate values by the random generator). In early simulations, completely random numbers were used. However, if these values were somehow too closely related, convergence speed could decrease significantly. Therefore, this more artificial coding scheme was chosen to implement the symbol codes. In speech recognition or computer vision applications, the Kohonen feature map has frequency, light intensity, and contrast ranges as input, all natural data input types. Here, we work with an artificial coding for words and sentences. Therefore, one can defend the choices made in applying this coding scheme. The final code for a symbol is assigned

randomly. The generation of different codes, is based on permutations of the base values. As far as possible, the complete coding space was used, so element codes were distributed equally throughout the element space (if a certain part of the map had a higher clustering density, than this was based on frequencies of occurrence and *not* on characteristics of the internal coding). Some examples can be found in the table below: *Internal Coding of the Elements.*

ELEMENTS AND CODES

#	Element	Code	Sensors					
0	NOUN-HUM	0	0.0000	0.0000	0.6667	1.0000	0.0000	0.6667	0.0000
1	VERB-EAT	1	0.0000	0.3333	0.3333	0.6667	0.3333	0.0000	0.0000
2	NOUN-FOOD	2	0.0000	0.6667	0.0000	0.3333	0.3333	0.6667	0.0000
3	VERB-PERCEPT	3	0.0000	0.6667	1.0000	0.0000	0.6667	0.0000	0.0000
4	NOUN-INANIM	4	0.0000	1.0000	0.3333	1.0000	0.6667	0.6667	0.0000
5	VERB-DESTROY	5	0.3333	0.0000	0.0000	0.6667	1.0000	0.0000	0.0000
6	NOUN-FRAG	6	0.3333	0.0000	1.0000	0.3333	1.0000	0.6667	0.0000
7	VERB-INTRAN	7	0.3333	0.3333	0.6667	0.3333	0.0000	0.0000	0.0000
8	VERB-TRAN	8	0.3333	0.6667	0.3333	0.0000	0.0000	0.6667	0.0000
9	VERB-AGPAT	9	0.3333	0.6667	1.0000	1.0000	0.3333	0.0000	0.0000
10	NOUN-ANIM	10	0.3333	1.0000	0.6667	0.6667	0.3333	0.6667	0.0000
11	NOUN-AGGRESS	11	0.6667	0.0000	0.3333	0.3333	0.6667	0.0000	0.0000

Internal Coding of the Elements

Semantotopic Map of Syntactic Structures (Type 1 Input)

After 380,000 learn cycles the following formation developed on the first map (upper part of *First and Second Map After 380.000 ... Constituents*). Every element represents a word fitting best in relation to the contents of the neuron. XXXXX means no reasonable mapping could be found. One can clearly distinguish the NOUN from the VERB part. Interesting are the neighbourhoods holding related syntactic-semantic objects like NOUN-ANIM/NOUN-INANIM, VERB-TRANS/VERB-INTRANS and VERB-DESTROY/NOUN-AGGRESS.

VERB-DESTROY	VERB-TRAN	VERB-INTRAN	VERB-AGPAT	VERB-AGPAT	VERB-AGPAT	VERB-PERCEPT	NOUN-HUM	VERB-EAT
VERB-DESTROY	VERB-TRAN	VERB-INTRAN	VERB-AGPAT	VERB-AGPAT	VERB-AGPAT	NOUN-HUM	NOUN-HUM	NOUN-HUM
VERB-DESTROY	VERB-TRAN	VERB-TRAN	NOUN-FOOD	VERB-AGPAT	VERB-AGPAT	VERB-AGPAT	NOUN-HUM	NOUN-INANIM
VERB-DESTROY	VERB-TRAN	NOUN-FOOD	NOUN-FOOD	NOUN-FOOD	VERB-AGPAT	NOUN-HUM	NOUN-HUM	NOUN-INANIM
VERB-DESTROY	NOUN-FOOD	NOUN-FOOD	NOUN-FOOD	NOUN-FOOD	NOUN-ANIM	NOUN-HUM	NOUN-HUM	NOUN-INANIM
NOUN-AGGRESS	NOUN-AGGRESS	NOUN-FOOD	NOUN-ANIM	NOUN-ANIM	NOUN-ANIM	NOUN-HUM	NOUN-HUM	NOUN-INANIM
NOUN-AGGRESS	NOUN-AGGRESS	NOUN-ANIM	NOUN-ANIM	NOUN-ANIM	NOUN-HUM	NOUN-HUM	NOUN-HUM	NOUN-INANIM
NOUN-AGGRESS	VERB-TRAN	NOUN-ANIM	NOUN-ANIM	NOUN-ANIM	NOUN-ANIM	NOUN-HUM	NOUN-HUM	NOUN-INANIM
VERB-DESTROY	VERB-TRAN	NOUN-ANIM	NOUN-ANIM	NOUN-ANIM	NOUN-ANIM	NOUN-HUM	NOUN-INANIM	NOUN-INANIM

xxxxxxxxxxxxxxxxxxxxxx	NOUN-HUM VERB-INTRAN	xxxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxx	NOUN-HUM VERB-AGPAT	NOUN-HUM VERB-PERCEPT NOUN-INANIM	xxxxxxxxxxxxxxxxxxx
NOUN-HUM VERB-PERCEPT NOUN-INANIM	xxxxxxxxxxxxxxxxxxx	NOUN-ANIM VERB-EAT NOUN-FOOD	NOUN-ANIM VERB-TRAN NOUN-ANIM
NOUN-HUM VERB-TRAN NOUN-HUM	NOUN-HUM VERB-EAT NOUN-FOOD	NOUN-HUM VERB-PERCEPT NOUN-INANIM	xxxxxxxxxxxxxxxxxxx

First and Second Map After 380.000 Learn Cycles with Syntactical Constituents

The second map (lower part of last figure) represents related contextual structures. At the left are sentences starting with a NOUN-HUM, at the right are the ones starting with NOUN-ANIM. Sentences holding VERB-EAT concentrate respectively at the left and the right side of these regions, resulting in a simple syntactic generalization. So, the net classifies and generalizes on semantic features as well as on syntactic structures.

One has to realize that the second map is not needed for context-sensitivity. A model without a second map (but with the recurrent fibres as described), processes sequential information (prediction of next element, grammatical correctness) just as well. However, the syntactic structures used in the disambiguation and structure assignment process do need information from the second map. The results of the prediction and grammar-checking process, improve by using additional information of the second map.

Even after so many cycles, the map still not converged completely (the state of self-organization from which there is no escape). However, results indicate that the net is converging towards such a state, which will probably be achieved after more than a million cycles. (In the current implementation, training the net with one million learn cycles takes more than a week of calculations. So far, this has only been carried out for the FSM simulations, which do reach a self-organized state).

Semantotopic Map from Unformatted Sentences of Type 2

The following table presents an example of the formation of a semantotopic map of sentences of the type {JOHN LOVES MARY} after 50,000 iterations. The map was formed from unformatted (flat) sentences passing by. The syntactic (or low-level semantic) category as well as the relation between categories can be derived from the position on the map.

Layer #0	man	man	monster	xxxxxxx	dragon	xxxxxx	break
	woman	move	smash	see	smell	eat	chase
	break	glass	smash	smell	break	break	chase
	break	break	glass	cookie	break	bread	xxxxx
	break	break	break	cookie	mouse	bread	xxxxx
	plate	glass	break	mouse	cat	cat	book
	plate	xxxxx	glass	break	mouse	rock	book

Feature Map after 50.000 Learn Cycles

Although there is not yet a clear separation on the map between nouns and verbs, the map holds substitutionally related elements in neighbouring regions. If similar maps are studied at vast time intervals, the map seems to alter periodically between convergence and divagation, shaking objects on the map, which results in a better organization after a while. This type of behaviour is characteristic for Kohonen feature maps. Here too, better results can be expected after more than a million learn cycles.

During the learning sessions, we counted the number of times a neuron was assigned *best match*. The result is shown below. Peaks in this distribution are spread over the map, showing the balance between recurrent and symbolic fibres. The high number 895 in the left corner is caused by the fact that the recurrent fibres of the first element in a string is set to zero, triggering an increased activation of this neuron. Here, only sentences of three words were used, resulting in relatively many sentence starts. If one counts these activations in the FSM simulations, two peaks are found, one for the start– and one for the end symbol. As sentences have a more varied length, these frequencies are more naturally distributed.

	HISTORY						
Layer #0	175	51	112	53	315	37	268
	70	100	72	77	58	5	165
	116	77	49	86	119	6	52
	67	28	56	62	78	24	66
	62	7	129	38	64	82	130
	118	76	76	15	15	251	15
	214	376	24	420	11	4	895

Activation Frequency of the Elements in the Map

String Prediction (Results of Type 3 Input)

To determine a grammatical lowerbound for the model, a Finite State Machine (FSM) similar to the one in [Cleeremans et al., 1989] was implemented to generate simple sentences. Although one cannot proof the theoretical equivalence between a recurrent neural net (RNN) and a FSM, experiments can indicate there is one. If the net accepts all strings generated by a FSM and rejects all the others, then the RNN probably implements at least a FSM: a grammatical lowerbound. This lowerbound is detemined experimentally and has no mathematical value what so ever. Prediction is possible in the model by feeding forward one element, determining the recurrent fibre and finding the best matches of this fibre on the first map. The symbolic parts of these neurons hold possible subsequent elements in the string. If the next element is predicted correctly, the string is accepted. Otherwise, the string would be rejected as being ungrammatical. By using a threshold in the matching process between the input and the weight of the recurrent fibres, one can determine the measure of correctness, an interesting value which provides an indication of the quality of the prediction. After learning the following results were obtained:

#	STRING	ELEMENT	PREDICTION OF NEXT ELEMENT
0	NOUN-HUM VERB-INTRAN	NOUN-HUM:	{VERB-INTRAN or VERB-AGPAT,}
		VERB-INTRAN:	{}
1	NOUN-HUM VERB-AGPAT NOUN-INANIM	NOUN-HUM:	{VERB-TRAN or VERB-AGPAT,}
		VERB-AGPAT:	{NOUN-INANIM, }
		NOUN-INANIM:	{}

Some String Prediction Examples

Grammatical Correctness

More than 1 million sequences were taught to the net. Afterwards, the net was tested with randomly generated sentences of which only 10% was grammatical. 99% of all grammatical strings were accepted, 1% rejected. Of all ungrammatical strings 0% was accepted. Main reason for the 1% failure of the grammatical strings were the long repetitions of *tttt* and *xxxxx*. Even after a million cycles one could define a sentence which would not be recognised (e.g. a sentence containing more than hundred *t*'s after each other). However, the longer the learning process, the better the performance. Below some examples of the simulations are shown: *Some String Acceptance and Rejection Examples*. Here too, a threshold function was used. If the error of the matches between the input values and the weights of the recurrent fibres grew above this value, the string was rejected; if the end of string symbol was reached before the threshold, the sentence was accepted. During the simulations, two different threshold models were used: one which compared the cumulative difference to a threshold, and one which compared the difference between input– and weight values of every character to a threshold. The last model worked much better then the first one.

#	STRING	ACCEPTED	GRAMMATICAL
0	btxse	YES	YES
1	bpttttttvve	YES	YES
2	bpvve	YES	YES
3	bse	NO	NO
4	bxe	NO	NO

Some String Acceptance and Rejection Examples

Disambiguation

Another linguistic task is word-sense disambiguation. Although hard to imagine, an organization can be achieved where all elements are *relatively* in such a position, that a statistical distribution is implemented which provides information on the plausibility of a certain meaning or structural function of a word, derivable from the regional part it activates. E.g., suppose one word has multiple meanings. This results in the formation of just one region on the map. However, this region is constructed of sub-regions for each different meaning of the word. If an additional shell keeps track of the position of words on the map, the exact meaning of a word within a context can be determined from the sub-region which is activated. If a structure has different meanings, the same algorithm can be applied to the second map, where a structure region is constructed out of ambiguous sub-structure regions. In this context it is important to realize that as long as the words are ordered relatively to each other, as many different relations as possible can be stored in a feature map. There is more than a two-dimensional ordering. As a matter of fact, this ordering is a projection of the multi-dimensional space where each dimension of the input vector (and weight vector) represents a dimension in that space. If one sees the ordering of words in this context, one might imagine a relative ordering capable of a disambiguation task. Especially the capability of the net to indicate a measure of correctness plays a significant role in this process. By calculating the difference between the input values and the weight values, one can determine the part of the region that corresponds best to the input values. The additional shell we mentioned above would assign a meaning to the region indicated, thus disambiguating the word.

The disambiguation task has only been investigated globally. The current model (simulation) has some disadvantages causing problems which can be solved better after some changes to the model as a whole. Most important is the insight about how one can disambiguate with a Kohonen feature map. The same holds for the linguistic task described in the next paragraph: the assignment of structures to constituents.

Assigning Structures to Constituents

Even more ambiguous is the task of assigning structures to constituents. The second map derives a distribution of the structures of the sentences presented to the first map in an unsupervised way. If an additional shell keeps track of the position of these regions after (and during) the processing of such sentences, it is possible to assign a structure to a sentence which is fed forward through the net. However, before the sentence structures are derived from scratch, a very long learn process must be gone through. Beside this disadvantage, one has to be aware that structures assigned to sentences are based on a statistical distribution of left-context sensitive words sequences. So, complicated structures might not be noticed, or are being confused with others. Mainly due to the long learn cycles, this task has not been evaluated in depth: only some simple (but successful) tests have been made. On the one hand, the statistical characteristics of the distribution can be helpful, on the other, the inexact match between a sentence presented and a structure on the map, can cause confusion. Once more, all this is based on the cooporation with an additional shell which knows exactly the locations of word meanings and sentence structures on the map. The feature map indicates the most plausible solution to a problem by activating the region holding the distributed information, but the final structure is assigned by the shell. It is impossible to store the explicit structure (E.g. Noun-verb-Noun) in the feature map. But who cares about these artificial notions anyway?

Overall Results

The overall results of the simulations are good. Although some side remarks can be made such as:

- Very long learning times
- The unstable character of the convergence process
- Lack of real insight in the convergence process, resulting in a situation where one never knows whether the map has finished converging, or that a local minimum is reached

On the other hand, the model seemed to be remarkably good in performing various linguistic tasks (although simulated in sometimes very simple ways) such as:

- Word classification
- Word class derivation
- Sentence structure derivation
- String prediction
- Grammar checking
- Disambiguation
- Structure assigning

Furthermore, the model worked in cooporation with a traditional shell which preprocessed the information and took care of the outputs generated by the model. This cooporation is not based on a synthesis between different mechanism within one model (such as a neural net extended with a stack), but two different mechanisms are incorporated in such a way that both do what they are best at. Therefore, the results are interesting enough to continue the research in the direction taken: self-organizing (unsupervised) temporal-processing models in natural-language processing inspired by neurological models of brain structures.

Discussion

By combining and re-interpreting work of [Ritter et al., 1989b], [Ritter et al., 1990], [Kangas, 1990] and [Elman, 1988] a model was developed which can derive a *semantotopic* map of language from unformatted strings. In this section, some important aspects of the model are discussed.

Grammatical Processing Capabilities

After theoretical analyses of the model, simulations based on examples from Elman's Simple Recurrent Network (SRN) [Elman, 1988], [Servan-Schreiber et al., 1988], [Cleeremans et al., 1989] and [Servan-Schreiber et al., 1988] resulted in similar results as obtained in their research. The model predicts new elements in a sequence, categorizes words according to their syntactical and semantical features, accepts finite-state grammars, derives contextual structures, categorizes these structures and generalizes over the information stored in both layers. Moreover, all features were learned by using a **un**supervised learning algorithm. The first map categorizes single words, the second map derives and categorizes contexts. A limitation of the model appears in the task of recognizing very long sequences (as indicated in the previous section). This lack in performance can be overcome by increasing learning-times. However, this research could not determine an upper bound.

Which grammar classes are recognisable by the net type is very important in this context. In [Tsung et al., 1989] it was shown that context feedback nets (Elman nets) could perform tasks which could not be solved by state feedback nets (Jordan nets). Next, [Cleeremans et al., 1989] proved that Finite State Grammars (FSG) are a lower bound for Elman nets. Recurrent fibres only cause a left-context sensitivity. More complicated grammars such as context sensitive grammars are not possible without introducing right-context sensitivity, which might only be possible by addition of buffering mechanisms (or short term memories). In [Powers, 1989] much psychological evidence for these mechanisms is given. Related research determining the grammatical capacities of recurrent nets can be found in [Giles et al., 1990], [Liu et al., 1990], [Sun et al., 1990a] and [Sun et al., 1990b] where context-free grammars are recognized by using higher order nets (recurrent fibres and state memories).

Convergence Properties

Although the model seems to converge, this process is very complex and difficult to monitor or influence. By decreasing the number of recurrent fibres, complexity decreases somewhat. Generally, there are two methods to perform this task: first one can reduce the dimension even more, second, instead of using fully interconnected layers, one might use Gaussian connections. The latter has the advantage of eliminating noise from irrelevant fibres (caused by their absence), which still are being used (although compressed) in the first option. On the other hand, convergence times for complicated FSG with nested sub-FSG in [Cleeremans et al., 1989] were about as long as the times spotted here, when taken in consideration that this model is unsupervised, convergence times can be called *within expectation.*

The balance between the number of recurrent and symbolic fibres is much to delicate in this model. If the norm or number of sensors is changed, convergence might not take place any longer. This instability must be overcome in future models.

For the moment, it is not clear when the model converges and when it doesn't. There are factors such as the decreasing learning rate and region size with obvious influences on the process. Furthermore, one can deduct the strict relation between the number of symbolic and recurrent fibres. But real insight into the process is lacking. In the single layer Kohonen feature map, it can be proven wether the model converges or not. More on this subject can be found in [Ritter et al., 1986], [Ritter et al., 1988], [Ritter et al, 1989a] and [Ritter, 1989].

In the model proposed here, two self-organizing models are influencing each other, resulting in a complicated mathematical process. It is useful to analyse this process in detail by mathematical means. However, research in this direction has not been carried out yet. One of the objectives of the development of future models is to take this aspect in consideration. See for instance [Simard et al., 1988], [Williams et al., 1988], [Williams et al., 1989a], [Williams et al., 1989b] and [Pineda, 1987], where in depth analysis of recurrent backpropagation is presented. The same analysis (be it in less detail and with more restrictions) must be possible for recurrent self-organization. Initial hints for the mathematical framework of this research can be found in [Sontag, 1990] and [Tanaka, 1990]

Temporal Processing

One of the main issues of the research carried out happened to be the implementation of temporal processing capabilities in (self-organizing) neural nets. In the introduction, it was mentioned that there are globally four different directions: dimension extension, windows, buffering and recurrent fibres. The more current research proceeded, the more recurrent fibres seem to lack the power needed for natural language processing. Of course, recurrent fibres are very important as a feed back function [Dell, 1985], but they are not powerful enough to take care of all temporal processes needed in a NLP system. Where the dimension extension and window mechanisms are not really plausible, flexible or elegant, the buffering mechanisms as proposed by [Kohonen et al., 1981] and indicated in [Miller, 1956], [Powers, 1989] and [Powers, 1991] show some additional advantages. However, the main problem is the implementation of such a system in a neural net without destroying the neural net computing paradigm and designing solutions which are very interesting from an engineering point of view, but not really from a linguistic or psychological one. Therefore, main efforts in future research shall be directed towards the development of more powerful temporal processing mechanisms.

Topological Maps of Language and Cognitive Maps

As mentioned, this model doesn't fit in the context natural language processing at first sight. How does one for instance defines a topological map of language? Or, how must one add contextual information in order to avoid organization on grounds of arithmetic features of the internal coding?

The question as stated on the interpretation of a *topological map of language* can be seen in the light of the task and performance of the two maps: the unsupervised syntactical– and semantical derivation– and categorization of elements and structures from bare sentences just passing by. An even more interesting way of looking at this question is in the usage of the feature map in the disambiguation task. As being a two-dimensional projection of a multiple dimensional input space, the Kohonen feature map forms a relative distribution of various word senses and sentence structures. This feature map can be compared with a topological map of language, where objects that are closely related (according to some features) have smaller distances to each other than to less related objects.

Most NLP models use the Kohonen map in cooporation with a backpropagation algorithm as an optimum clustering device. Here, it is shown that Kohonen models are capable of processing symbolic data even in tasks other than conceptualizing. The question raised earlier on the interpretation of a topological map of language might, in the light of the effects on the second map, be answered by comparing its function with a part of the cognitive maps described in [Stolcke, 1990] and elsewhere in literature [Lakoff, 1988] [Chrisley, 1990].

Related Work

Although the Kohonen self-organizing model is just an efficient statistical classifier, it is capable of deriving semantical features of symbolic data, as long as data is presented in its proper context. The same feature of neural nets can be seen in work carried out by [Miikulainen et al., 1988a, 1988b], [St. John et al., 1988a], and [St. John et al., 1988b, 1990], where generalization over context resulted in the automatic derivation of semantic (micro-) features. This ability of neural nets in general cannot be found in classical symbolic AI, without the addition of complex procedural modules.

The lack of good definitions of recurrent mechanism in self-organizing systems leaves plenty of space for further research towards other models. Therefore, recent work done in recurrent or spatio-temporal self-organization shows a lot of variability in theory, architecture and implementation [Kangas, 1990], [Kangas et al., 1990], [Koikkalainen et al., 1990], [Samaranbunda et al., 1990], [Silverman, 1988], [Stotzka et al., 1990], [Tavan et al., 1990], [Thacker et al., 1990], [Yen et al., 1990]. Other work by Linsker and the even more biologically inspired Neuronal Group Selection theory of Reeke & Edelman might also be suited to implement linguistic phenomena [Edelman, 1987], [Reeke et al., 1988] [Reeke et al., 1990]. The main problem with all these unsupervised models is the complexity of the simulations and the less developed foundations, making NLP application research quite tricky. Various hybrid solutions try to overcome the disadvantages of self-organizing models. A possible solution is to use a self-organizing feature map to discover the features in the learn set, and back-propagate between these maps to learn and generalize between input and output pairs (or between input patterns and regions on the map). The efficient back-propagation algorithm then limits the complexity and uses known mechanisms, like recurrent connections, to implement complex phenomena. More on these solutions can be found in [Hrycej et al., 1989], [Hrycej et al., 1990] and [Gersho et al., 1990]. Other hybrid solutions where either self-organizing and backpropagating nets, or self-organizing nets and symbolic techniques or backpropagation neural nets and symbolic methods are combined, are described in [Dolan et al., 1987], [Dyer, 1988], [Dyer, 1990], [Honavar et al., 1989], [Honavar et al., 1990], [Jain et al., 1990]

Future Work

Current work in progress concentrates on the implementation of other linguistic tasks such as disambiguation, parsing, language acquisition and the derivation of simple analogies. Beside these applications, extensions to the model are being implemented based on a stronger synthesis between cognitive science and neuroscience. Gaussian interconnections, additional layers and more Hebbian learn rules are an indication of the direction chosen. Furthermore, other mechanisms of processing temporal information must be developed which are more powerful than recurrent connections as used in this research, but not as unelegant as dimension extensions or window mechanisms.

Lately, this relation between self-organizing feature maps (which are computationally efficient with respect to the fact that they are self-organizing) and more biological which are based on neuroscientific brain research, but very hard to simulate with computer implementations, are getting more and more attention. By extending the Kohonen formalism towards multi-layer models with some variations in neuron type and learning rules, one can use an already evaluated theory in new models. Experiments in this direction are found in [Erdi, 1990] and [Ojemann, 1983].

So, future work involves model extensions and developments in the spirit of the Neuronal Group Selection theory [Edelman, 1987] or inspired by a stronger synthesis between cognitive science and neuroscience [Eimas et al., 1990]. In addition, a more fundamental model for (connectionist) language acquisition [Weber et al., 1990] [Feldman et al., 1990] can be used to evaluate acquisition performance in other connectionist models for NLP.

Conclusions

Finally, a number of conclusions on recurrent self-organization in Natural Language processing will be summarized to provide the reader with a brief overview on the research carried out.

Temporal Processing

- Recurrent fibres implement at least a FS grammar. The types of grammars, the length of the sequences and other properties of these models are quite unknown yet. Future research must provide a better insight in these aspects.

- Recurrent connections alone are not powerful enough. However, neural structures capable of buffering and other sophisticated mechanisms (observed in humans) are not developed yet.

Self-Organization vs Backpropagation

- Self-organizing techniques can overcome some of the disadvantages of the back-propagation algorithm. The main problem with these self-organizing models is the exponentially increasing complexity. Especially the addition of recurrent fibres enlarges the time required to process the input data. One might accept these disadvantages, because the limitation of back propagation (e.g. the need to learn input/output pairs, the pre-wiring of lateral inhibition, the definition of micro-features and the need to pass the entire learn-set again after addition of new elements) are even worse.

- On the other hand, one might maintain that self-organization is still too simple and should be extended towards more Hebbian and Evolutionary models. To implement application research in these directions is very tricky due to the undeveloped character of this field. However, pioneering work can be observed in the literature.

Temporal (Recurrent) Self-Organization

- Recurrent self-organization is still in its early development. This is mainly caused by the limited knowledge of self-organization as a whole. Additional research can provide the insights needed here. More interesting is a recurrent mechanism which is equipped with some sort of buffering mechanism, making it sensitive for left as well as right context.

- The main problem in (current) recurrent self-organization is the lack of insight into the convergence process. This should definitely be worked on in the near future, if this direction is continued .

- Simulations are quite reasonable with respect to recurrent backpropagation. However, learning times which take more than a week are never acceptable, therefore more efficient implementations or more powerful computers should be used in new experiments.

Unsupervised Language Learning

- Although limited, a completely autonomous model for the derivation of context dependent semantics is developed (or in other words, semantic features are derived by generalizing over contexts). The exact properties are not known yet, but this semantics is just the semantics that is hard to obtain by logic and other commonly used semantic techniques. Just therefore the results are interesting enough to continue further research.

- Possible extensions might concern as well other, more powerful and complicated models, as well as more thoroughly defined examples in e.g. language acquisition as proposed in [Feldman et al., 1990] and [Weber et al., 1990].

Hybrid Solutions

- All the simulation results were obtained from the cooporation between a recurrent self-organizing neural net simulator and a traditional shell which prepared and interpreted the information from the neural net by remembering which position stored which information. At this moment, it is impossible to store linguistic structures in a neural net without such an interface to the outside world. Therefore additional research towards the efficient use of (other) hybrid solutions is definitely needed.

Cognitive Neuroscience

- As a result of this research, the author feels more and more attracted to the idea that the only way to reach real achievements in connectionist natural language processing is by combining knowledge from mathematics, cognition, neural sciences and linguistics. Some call it cognitive neuroscience, others neurolinguistics, or (cognitive) cybernetics. Whatever the name, the direction to be taken is clear. By extending the Kohonen formalism, a first step is made. Future research continues in this direction in order to achieve a continally growing synthesis between cognition and neural sciences.

Acknowledgements

The author wishes to thank Remko Scha for many fruitful discussions over the last two years. Bob Allen, James Anderson, Brian Bartell, David Chalmers, Gary Cottrell, Renate Deffner, Hans Henseler, Susan Hollbach-Weber, Vasant Honavar, Karen Kukich, Catherine Myers, David Powers, Jurgen Schmidhuber, Eduardo Sontag, Andreas Stolcke and Fu-Sheng Tsung for exchanging views or providing information on the issues discussed, and the members of the Discouse Processing and Data Oriented Parsing workshop (DIPDOP) of the University of Amsterdam for their early comments on this work.

The research presented here was reported on before in [Scholtes, 1991a] up to [Scholtes, 1991g].

References

[Allen, 1990]: Allen, R.B. (1990). Connectionist Language Users. Technical Report, Bell Communications Research.

[Carpenter et al., 1988]: Carpenter, G.A. and Grossberg, S. (1988). The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network. IEEE Computer, Special Issue on Artificial Neural Systems, Vol. 21, nr. 3, pp. 77-88.

[Chrisley, 1990]: Chrisley, R.L. (1990). Cognitive Map Construction and Use: A Parallel Distributed Processing Approach. In: Connectionist Models: Proc. of the 1990 Summer School, pp. 287-300. Morgan-Kaufman Publishers.

[Cleeremans et al., 1989]: Cleeremans, A., Servan-Schreiber, D. and McClelland, J.L. (1989). Finite State Automata and Simple Recurrent Networks. Neural Computation, Vol. 1, No. 3, pp. 372-381.

[Dell, 1985]: Dell, G.S. (1985). Positive Feedback in Hierarchical Models: Applications to Language Production. Cognitive Science, Vol. 9, pp. 3-23.

[Dolan et al., 1987]: Dolan, C.P. and Dyer, M.G. (1987). Symbolic Schemata, Role Binding, and the Evolution of Structure in Connectionist Memories. Proceedings of the IEEE 1st INNC San Diego, pp. 287-298.

[Dyer, 1988]: Dyer, M.G. (1988). Symbolic NeuroEngineering for Natural Language Processing: A Multilevel Research Approach. TR UCLA-AI-88-14, UCLA, August 1988.

[Dyer, 1990]: Dyer, M.G. (1990). Distributed Symbol Information and Processing in Connectionist Networks. Journal of Expt. Artif. Intell., Vol. 2, pp. 215-239.

[Edelman, 1987]: Edelman, G.M. (1987). Neural Darwinism: The Theory of Neuronal Group Selection. Basic Books Inc.

[Edelman, 1989]: Edelman, G.M. (1989). The Remembered Present: A Biological Theory of Consciousness. Basic Books Inc.

[Eimas et al., 1990]: Eimas, P.D. and Galaburda, A.M. (Eds.) (1990). Neurobiology of Cognition. A Cognition Special Issue, MIT Press.

[Elman, 1988]: Elman, J.L. (1988). Finding Structure in Time. Technical Report CRL 8801, Center for the Research of Language, UCSD.

[Erdi, 1990]: Erdi, P. (1990). Self-Organization in the Nervous System: Network Structure and Stability. TR-KFKI-1990-60/H, Hungarian Academy of Sciences, Central Research Institute for Physics, Budapest.

[Feldman et al., 1990]: Feldman, J.A., Lakoff, G., Stolcke A. and Weber, S.H. (1990). Miniature Language Acquisition: A Touchstone for Language Acquisition. TR-90-009, March 1990, ICSI Berkeley, CA.

[Gersho et al., 1990]: Gersho, M. and Reiter, R. (1990). Information Retrieval using a Hybrid Multi-Layer Neural Network. Proceedings of the IJCNN, San Diego, June 17th-21th, 1990, Vol. 2, pp. 111-117.

[Giles et al., 1990]: Giles, C.L., Sun, G.Z., Chen, H.H., Lee, Y.C. and Chen, D. (1990). Higher Order Recurrent Networks and Grammatical Inference. In: Advances in NIPS (D.S. Touretsky, Editor), Vol. 2, pp. 380-387. Morgan Kaufmann Publishers.

[Goldberg et al., 1988]: Goldberg, D.E. and Holland, J.H. (Editors) (1988). Machine Learning (Special Issue on Genetic Algorithms). Vol. 3, Nos. 2-3.

[Graubard, 1988]: Graubard, S.R. (Editor) (1988). The Artificial Intelligence Debate. False Starts, Real Foundations. MIT Press.

[Grossberg, 1980]: Grossberg, S. (1980). How Does a Brain Build a Cognitive Code?. Psychological Review, Vol. 87, pp. 1-51.

[Grossberg, 1988]: Grossberg, S. (1988). Nonlinear Neural Nets. Neural Networks, Vol. 1, Nr. 1, pp. 17-61.

[Hebb, 1949]: Hebb, D.O. (1949). The Organization of Behavior: A Neuropsychological Theory. John Wiley & Sons, New York.

[Hemani, et al., 1990]: Hemani, A. and Postula, A. (1990). Scheduling by Self-Organization. Proceedings of the IJCNN, Washington, 1990, Vol. 2, pp. 543-546.

[Hodges et al., 1990]: Hodges, R.E. and Wu, C.H. (1990). A Method to Establish an Autonomous Self-Organizing Feature Map. Proceedings of the IJCNN, Washington, 1990, Vol. 1, pp. 517-520.

[Holland, 1975]: Holland, J.H. (1975). Adaptation in Natural and Artificial Systems. The University of Michigan Press.

[Honavar et al., 1989]: Honavar, V. and Uhr, L. (1989). Generation, Local Receptive Fields and Global Convergence Improve Perceptual Learning in Connectionist Networks. Proceedings of the 11th IJCAI, Vol. 1, pp. 180-185.

[Honavar et al., 1990]: Honavar, V. and Uhr, L. (1990). Symbol Processing Systems, Connectionist Networks, and Generalized Connectionist Networks. TR#90-24, Department of Computer Science, Iowa State University, Ames, Iowa, December, 1990.

[Hrycej, 1989]: Hrycej, T. (1989). Unsupervised Learning by Backward Inhibition. Proceedings of the 11th IJCAI, pp. 170-175.

[Hrycej, 1990]: Hrycej, T. (1990). Self-Organization by Delta Rule. Proceedings of the IJCNN, San Diego, June 17th-21th, 1990, Vol. 2, pp. 307-312.

[Jain et al., 1990]: Jain, A.N. and Waibel, A.H. (1990). Incremental Parsing by Modular Recurrent Connectionist Networks. In: Advances in Neural Information Processing Systems (D.S. Touretsky, Editor), Vol. 2, pp. 364-371. Morgan Kaufmann Publishers.

[Kangas et al., 1990]: Kangas, J.A., Kohonen, T.K. and Laaksonen, J.T. (1990). Variants on Self-Organizing Maps. IEEE Transactions on Neural Networks, Vol. 1, No. 1, pp. 93-99.

[Kangas, 1990]: Kangas, J. (1990). Time-Delayed Self-Organizing Maps. Proceedings of the IJCNN, San Diego, June 17th-21th, 1990, Vol. 2, pp. 331-336.

[Kohonen et al., 1981]: Kohonen, T., Oja, E. and Lehtio, P. (1981). Storage and Processing of Information in Distributed Associative Memory Systems. In: Parallel Models of Associative Memory (G.E. Hinton & J.A. Anderson, Eds.), Lawrence Erlbaum.

[Kohonen, 1982a]: Kohonen, T. (1982). Analysis of a Simple Self-Organizing Process. Biological Cybernetics, Vol. 44, pp. 135-140.

[Kohonen, 1982b]: Kohonen, T. (1982). Clustering, Taxonomy, and Topological Maps of Patterns. Proceedings of the 6th International Conference on Pattern Recognition, pp. 114-128.

[Kohonen, 1982c]: Kohonen, T. (1982). Self-Organized Formation of Topologically Correct Feature Maps. Biological Cybernetics, Vol. 43, pp. 59-69.

[Kohonen, 1984]: Kohonen, T. (1984). Self-Organization and Associative Memory. Springer-Verlag.

[Kohonen, 1988]: Kohonen, T. (1988). An Introduction to Neural Computing. Neural Networks, Vol. 1, nr. 1, pp. 3-16.

[Kohonen, 1990a]: Kohonen, T. (1990). Some Practical Aspects of the Self-Organizing Maps. Proceedings of the IJCNN, Washington, 1990, Vol. 2, pp. 253-256.

[Kohonen, 1990b]: Kohonen, T. (1990). The Self-Organizing Map. Proc. of the IEEE, Vol. 78, pp. 1464-1480.

[Koikkalainen et al., 1990]: Koikkalainen, P. and Oja, E. (1990). Self-Organizing Hierarchical Feature Maps. Proceedings of the IJCNN, San Diego, June 17th-21th, 1990, Vol. 2, pp. 279-284.

[Lakoff, 1988]: Lakoff, G. (1988). A Suggestion for a Linguist with Connectionist Foundations. Proceedings of the Connectionist Models Summer School, Carnegie Mellon University, pp. 301-314.

[Linsker, 1988]: Linsker, R. (1988). Self-Organization in a Perceptual Network. IEEE Computer, Special Issue on Artificial Neural Systems, Vol. 21, nr. 3, pp. 105-117.

[Liu et al., 1990]: Liu, Y.D., Sun, G.Z., Chen, H.H. and Lee, Y.C. (1990). Grammatical Inference and Neural Network State Machines. Proceedings of the IJCNN, Washington 1990, Vol. 1, pp. 285-288.

[Malsberg, 1973]: Malsberg, Chr. von der (1973). Self-Organization of Orientation Sensitive Cells in the Striate Cortex. Kybernetik, Vol. 14, pp. 85-100.

[McClelland et al., 1986a]: McClelland, J.L. and Rumelhart, D.E. (Eds.) (1987). Explorations in the Microstructure of Cognition. Volume 2: Psychological and Biological Models. MIT Press.

[McClelland et al., 1986b]: McClelland, J.L. and Rumelhart, D.E. (1986). Explorations in Parallel Distributed Processing. A Handbook of Models, Programs, and Exercises. MIT Press.

[Miikkulainen et al., 1988a]: Miikkulainen, R. and Dyer, M.G. (1988). Encoding Input/Output Representations in Connectionist Cognitive Systems. Proceedings of the Connectionist Models Summer School, Carnegie Mellon University, pp. 347-356.

[Miikkulainen et al., 1988b]: Miikkulainen, R. and Dyer, M.G. (1988). Forming Global Representations with Extended Back Propagation. Proceedings of the 2nd IEEE ICNN, San Diego.

[Miller, 1956]: Miller, A.M. (1956). The Magical Number Seven, Plus or Minus Two: Some Limits to our Capacity for Processing Information. Psychological Review, Vol. 63, pp. 81-97.

[Ojemann, 1983]: Ojemann, G.A. (1983). Brain Organization for Language from the Perspective of Electrical Stimulating Mapping. Behav. Brain Science, pp. 189-230.

[Pineda, 1987]: Pineda, F.J. (1987). Generalization of Back Propagation to Recurrent Neural Networks. Psychological Review Letters, Vol. 59, nr. 19, pp. 2229-2232.

[Powers et al., 1989]: Powers, D.M.W. and Turk, C.C.R. (1989). Machine Learning of Natural Language. Springer-Verlag.

[Powers, 1991]: Powers, D.M.W. (1991). Experiments in Unsupervised Language Learning. Submitted for Publication.

[Reeke et al., 1988]: Reeke, G.N. and Edelman, G.M. (1988). Real Brains and Artificial Intelligence. In: The AI Debate. False Starts, Real Foundations (R. Graubard, Editor), pp. 143-174. MIT Press, Cambridge.

[Reeke et al., 1990]: Reeke, G.N., Sporns, O. and Edelman, G.M. (1990). Synthetic Neural Modeling: The "Darwin" Series of Recognition Automata. Proceedings of the IEEE, Vol. 78, No. 9, pp. 1498-1530.

[Ritter et al., 1986]: Ritter, H. and Schulten, K. (1986). On the Stationary State of Kohonen's Self-Organizing Sensory Mapping. Biological Cybernetics, Vol. 54, pp. 99-106.

[Ritter et al., 1988]: Ritter, H. and Schulten, K. (1988). Kohonen's Self-Organizing Maps: Exploring their Computational Capabilities. Proc. IEEE ICNN, San Diego, Vol. 1, pp. 109-116.

[Ritter et al., 1989a]: Ritter, H. and Schulten, K. (1989). Convergency Properties of Kohonen's Topology Conserving Maps: Fluctuations, Stability and Dimension Selection. Biological Cybernetics, Vol. 60, pp. 59-71.

[Ritter et al., 1989b]: Ritter, H. and Kohonen, T. (1989). Self-Organizing Semantical Maps. Biological Cybernetics, Vol. 61, pp. 241-254.

[Ritter et al., 1990]: Ritter, H. and Kohonen, T. (1990). Learning "Semantotopic Maps" from Context. Proceedings of the IJCNN, Washington, 1990, Vol. 1, pp. 23-26.

[Ritter, 1989]: Ritter, H. (1989). Combining Self-Organizing Maps. Proceedings of the IJCNN, Washington, 1989, Vol. 2, pp. 499-502.

[Rubner et al., 1990]: Rubner, J. and Schulten, K. (1990). Development of Feature Detectors by Self-Organization. Biological Cybernetics, Vol. 62, pp. 193-199.

[Rumelhart et al., 1985]: Rumelhart, D.E. and Zipser, D. (1985). Feature Discovery by Competitive Learning. Cognitive Science, Vol. 9, pp. 75-112.

[Rumelhart et al., 1986c]: Rumelhart, D.E. and McClelland, J.L. (Eds.) (1986). Explorations in the Microstructure of Cognition. Volume 1: Foundations. MIT Press.

[Samarabandu, et al., 1990]: Samarabandu, J.K. and Jakubowics, O.G. (1990). Principles of Sequential Feature Maps in Multi-level Problems. Proceedings of the IJCNN, Washington, 1990, Vol. 2, pp. 683-686.

[Scholtes, 1990]: Scholtes, J.C. (1990). Trends in Neurolinguistics. Proceedings of the IEEE Symposium on Neural Networks, 21st June 1990, Delft, Netherlands, pp. 69-86.

[Scholtes, 1991a]: Scholtes, J.C. (1991). Kohonen's Self-Organizing Map Applied Towards Natural Language Processing. Proceedings of the CUNY 1991 Conference on Sentence Processing, Rochester, 12th-14th May 1991.

[Scholtes, 1991b]: Scholtes, J.C. (1991). Kohonen's Self-Organizing Map in Natural Language Processing. Proceedings of the SNN Symposium, Nijmegen, Netherlands, 1st and 2nd May 1991.

[Scholtes, 1991c]: Scholtes, J.C. (1991). Learning Simple Semantics by Self-Organization. Proceedings of the AAAI Symposium on Connectionist Learning of Natural Language, Palo Alto, 26st-29th March 1991.

[Scholtes, 1991d]: Scholtes, J.C. (1991). Learning Simple Semantics by Self-Organization. Proceedings of the AAAI Symposium on Machine Learning of Natural Language and Ontology, Palo Alto, 26st-29th March 1991.

[Scholtes, 1991e]: Scholtes, J.C. (1991). Recurrent Kohonen Self-Organization in Natural Language Processing. Proceedings of the ICANN, Helsinki, 26st-29th June 1991.

[Scholtes, 1991f]: Scholtes, J.C. (1991). Self-Organized Language Learning. Proceedings of the Annual Conference Cybernetics: Its Evolution and Its Praxis, Amherst, July 1991.

[Scholtes, 1991g]: Scholtes, J.C. (1991). Using Extended Kohonen-Feature Maps in a Language Acquisition Model. Proceedings of the 2nd Australian Conference on Neural Networks, Sydney, February 2nd-4th 1991.

[Schyns, 1990a]: Schyns, P.G. (1990). A Modular Neural Network Model of the Acquisition of Category Names in Children. In: Connectionist Models: Proc. of the 1990 Summer School, pp. 228-235. Morgan-Kaufman Publishers.

[Schyns, 1990b]: Schyns, P.G. (1990). Expertise Acquisition Through the Refinement of a Conceptual Representation in a Self-Organizing Architecture. Proc. of the IJCNN, Washington DC, pp. 236-240.

[Servan-Schreiber et al., 1988]: Servan-Schreiber, D., Cleeremans, A. and McClelland, J.L. (1988). Encoding Sequential Structure in Simple Recurrent Networks. TR CMU-CS-88-183, Carnegie-Mellon University.

[Servan-Schreiber et al., 1989]: Servan-Schreiber, D., Cleeremans, A. and McClelland, J.L. (1989). Learning Sequential Structure in Simple Recurrent Networks. In: Advances in NIPS (D.S. Touretsky, Editor), Vol. 1, pp. 643-652. Morgan Kaufmann Publishers.

[Silverman, 1988]: Silverman, R.H. (1988). Time-Sequential Self-Organization of Hierarchical Neural Networks. In: Neural Information Processing Systems (D.Z. Anderson, Editor), pp. 709-714. American Institute of Physics.

[Simard et al., 1988]: Simard, P.Y., Ottaway, M.B. and Ballard, D.H. (1988). Analysis of Recurrent Backpropagation. Proceedings of the Connectionist Models Summer School, Carnegie Mellon University, pp. 103-113.

[Sontag, 1990]: Sontag, E.D. (1990). Feedback Stabilization Using Two-Hidden Layer Nets. Report SYCON-90-11, Department of Mathematics, Rutgers University, New Brunswick, NJ.

[St. John et al., 1988a]: St. John, M.F. and McClelland, J.L. (1988). Applying Contextual Constraints in Sentence Comprehension. Proceedings of the Connectionist Models Summer School, Carnegie Mellon University, pp. 338-346.

[St. John et al., 1988b]: St. John, M.F. and McClelland, J.L. (1988). Applying Contextual Constraints in Sequence Comprehension. Proceedings of the Cognitive Science Society, Montreal, 1988, pp. 26-35.

[St. John et al., 1990]: St. John, M.F. and McClelland, J.L. (1990). Learning and Applying Contextual Constraints in Sentence Comprehension. Artificial Intelligence, In Press.

[Stolcke, 1990]: Stolcke, A. (1990). Learning Feature-Based Semantics with Simple Recurrent Networks. Technical Report TR-90-015, April 1990, ICSI Berkeley, CA.

[Stotzka et al., 1990]: Stotzka, R. and Manner, R. (1990). Self-Organization in a Linear Multi-Layered Feed forward Neural Network. Proceedings of the IJCNN, Washington, 1990, Vol. 1, pp. 126-129.

[Sun et al., 1990a]: Sun, G.Z., Chen, H,H,, Lee, Y.C. and Giles, C.L. (1990). Recurrent Neural Nets, Hidden Markov Models and Stochastic Grammars. Proceedings of the IEEE IJCNN, San Diego, July 1990, Vol. 1, pp. 729-734.

[Sun et al., 1990b]: Sun, G.Z., Chen, H.H., Giles, C.L., Lee, Y.C. and Chen, D. (1990). Connectionist Pushdown Automata that Learn Context-Free Grammars. Proceedings of the IJCNN, Washington, 1990, Vol. 1, pp. 577-580.

[Tanaka, 1990]: Tanaka, S. (1990). Theory of Self-Organization of Cortical Maps: Mathematical Framework. Neural Networks, Vol. 3, No. 6, pp. 625-640.

[Tavan et al., 1990]: Tavan, P., Grubmüller, H. and Kühnel, H. (1990). Self-Organization of Associative Memory and Pattern Classification: Recurrent Signal Processing on Topological Feature Maps. Biological Cybernetics, Vol. 64, pp. 95-105.

[Thacker et al., 1990]: Thacker, N.A. and Mayhew, J.E.W. (1990). Designing a Layered Network for Context Sensitive Pattern Classification. Neural Networks, Vol. 3, No. 3, pp. 291-299.

[Tsung et al., 1989]: Tsung, F.-S. and Cottrell, G.W. (1989). A Sequential Adder Using Recurrent Networks. Proceedings of the IJCNN, Washington D.C., June 1989.

[Weber et al., 1990]: Weber, S.H. and Stolcke, A. (1990). L0: A Testbed for Miniature Language Acquisition. TR-90-010, July 1990, ICSI Berkeley.

[Werner et al., 1990]: Werner, G.M. and Dyer, M.G. (1990). Evolution of Communication in Artificial Organisms. TR-AI-90-06, UCLA.

[Williams et al., 1988]: Williams, R.J. and Zipser, D. (1988). A Learning Algorithm for Continually Running Fully Recurrent Networks. Technical Report ICS Report 8805, UCSD.

[Williams et al., 1989a]: Williams, R.J. and Zipser, D. (1989). Experimental Analysis of the Real-Time Recurrent Learning Algorithm. Connection Science, Vol. 1, No. 1, pp. 87-111.

[Williams et al., 1989b]: Williams, R.J. and Zipser, D. (1989). A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. Neural Computation, Vol. 1, pp. 270-280.

[Yen et al., 1990]: Yen, M.M., Blackburn, M.R. and Nguyen, H.G. (1990). Feature Maps based Weight Vectors for Spatiotemporal Pattern Recognition with Neural Nets. Proceedings of the IJCNN, San Diego, June 17st-21st, 1990, Vol. 2, pp. 149-155.